

PAT 463/563 (Fall 2026)

Music & AI

Lecture 10: Convolutional Neural Networks

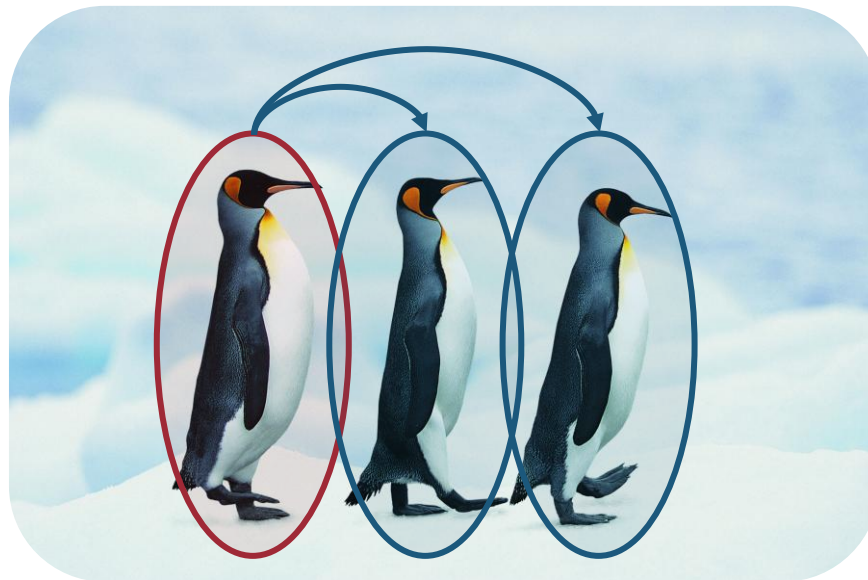
Instructor: Hao-Wen (Herman) Dong

Convolutional Neural Networks (CNNs)

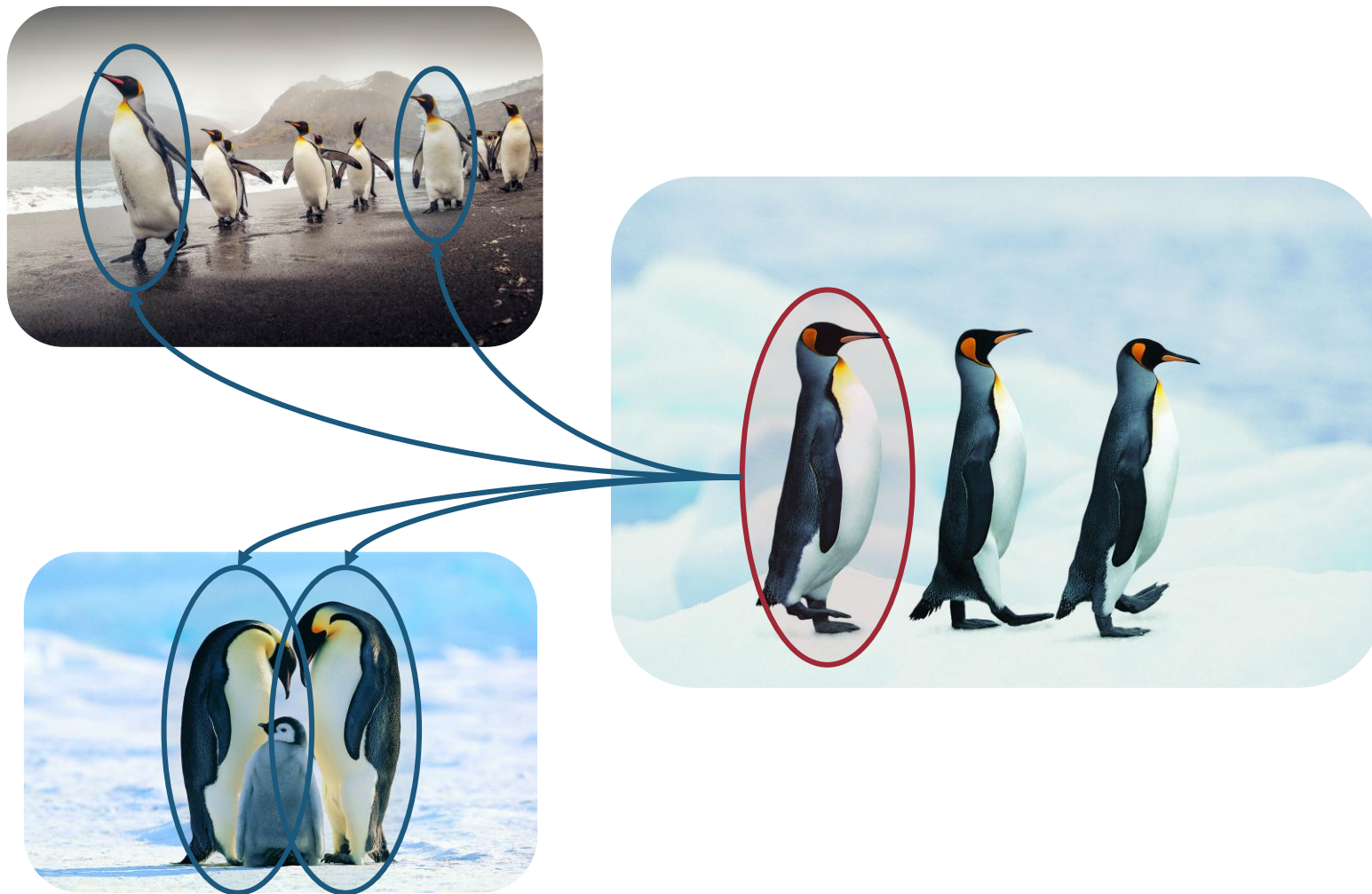
Convolutional Neural Networks (CNNs)

- **Intuition**: Learn **reusable local pattern detector**
- Widely used in **computer vision**
- Also used for music and audio
 - Representing music as **piano rolls**
 - Representing audio as **spectrograms**

| Reusable Pattern Detectors



Reusable Pattern Detectors



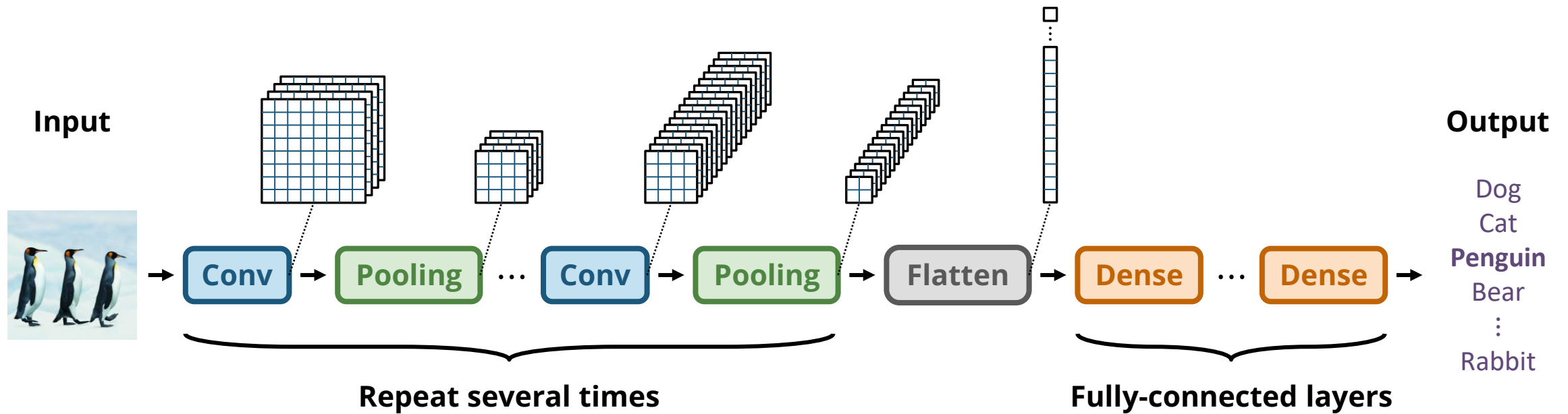
Reusable Pattern Detectors



Reusable Pattern Detectors



Convolutional Neural Network (CNNs)



2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

*

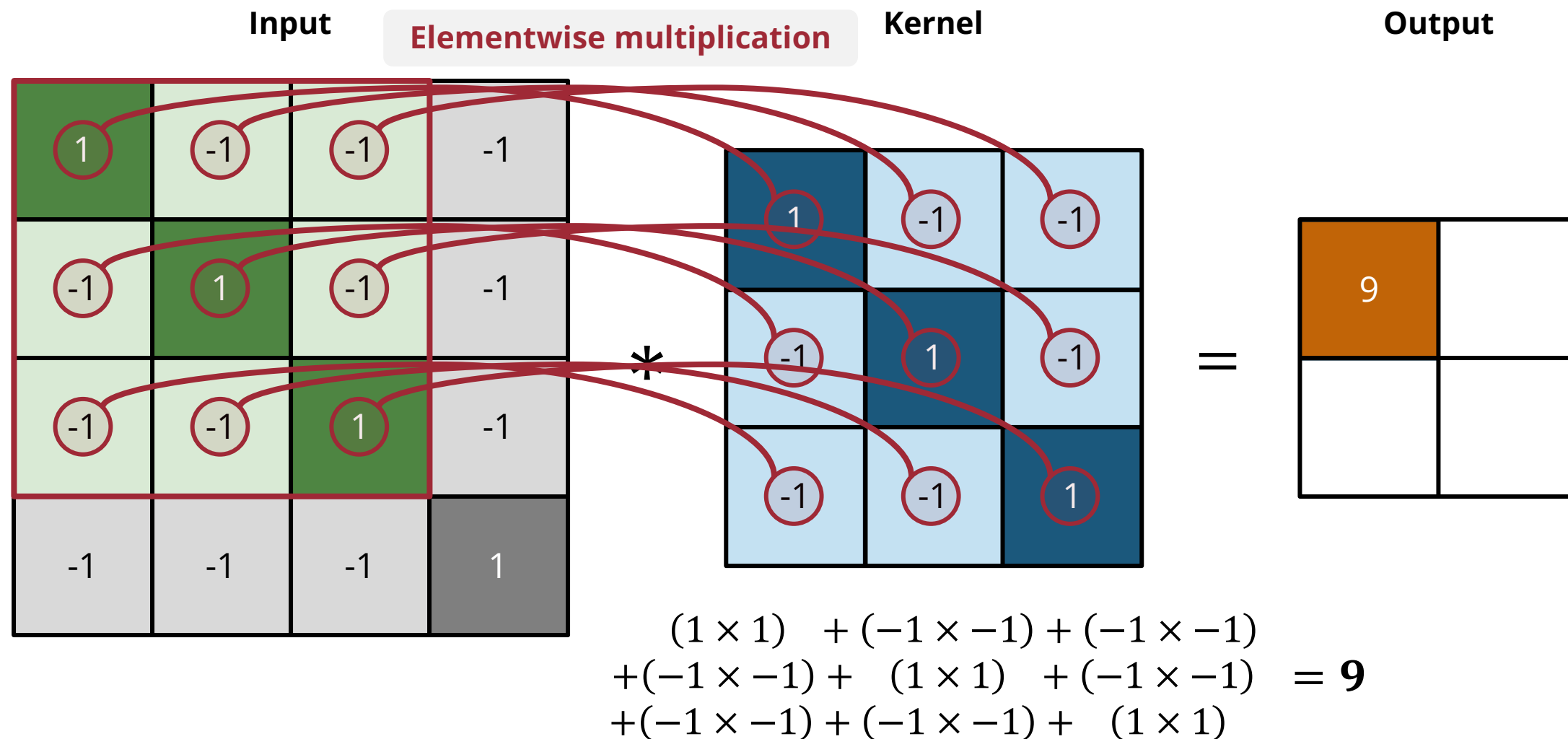
Kernel

1	-1	-1
-1	1	-1
-1	-1	1

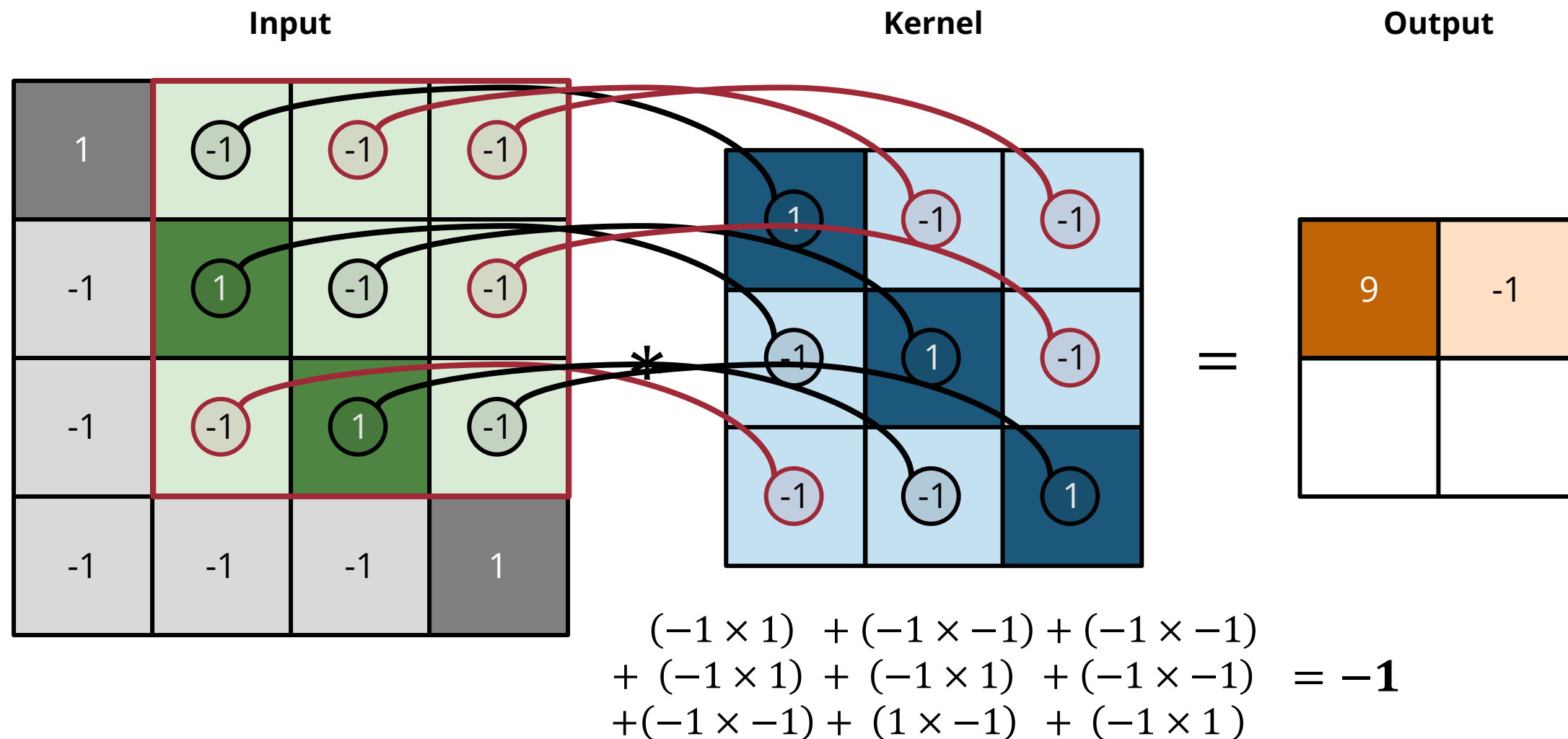
=

Output

2D Convolution



2D Convolution



2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

Kernel

1	-1	-1
-1	1	-1
-1	-1	1

*

=

Output

9	-1
-1	

$$\begin{aligned} & (-1 \times 1) + (1 \times -1) + (-1 \times -1) \\ & + (-1 \times -1) + (-1 \times 1) + (1 \times -1) \\ & + (-1 \times -1) + (-1 \times -1) + (-1 \times 1) \end{aligned} = -1$$

2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

Kernel

1	-1	-1
-1	1	-1
-1	-1	1

*

=

Output

9	-1
-1	9

$$\begin{aligned} & (1 \times 1) + (-1 \times -1) + (-1 \times -1) \\ & + (-1 \times -1) + (1 \times 1) + (-1 \times -1) \\ & + (-1 \times -1) + (-1 \times -1) + (1 \times 1) = 9 \end{aligned}$$

2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

*

Kernel

1	-1	-1
-1	1	-1
-1	-1	1

=

Output

9	
	9

High activation when
the local pattern is
close to the kernel

2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

*

Kernel

1	-1	-1
-1	1	-1
-1	-1	1

=

Output

	-1
-1	

**Low activation when
the local pattern
differs from the kernel**

2D Convolution

Input

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

*

Kernel

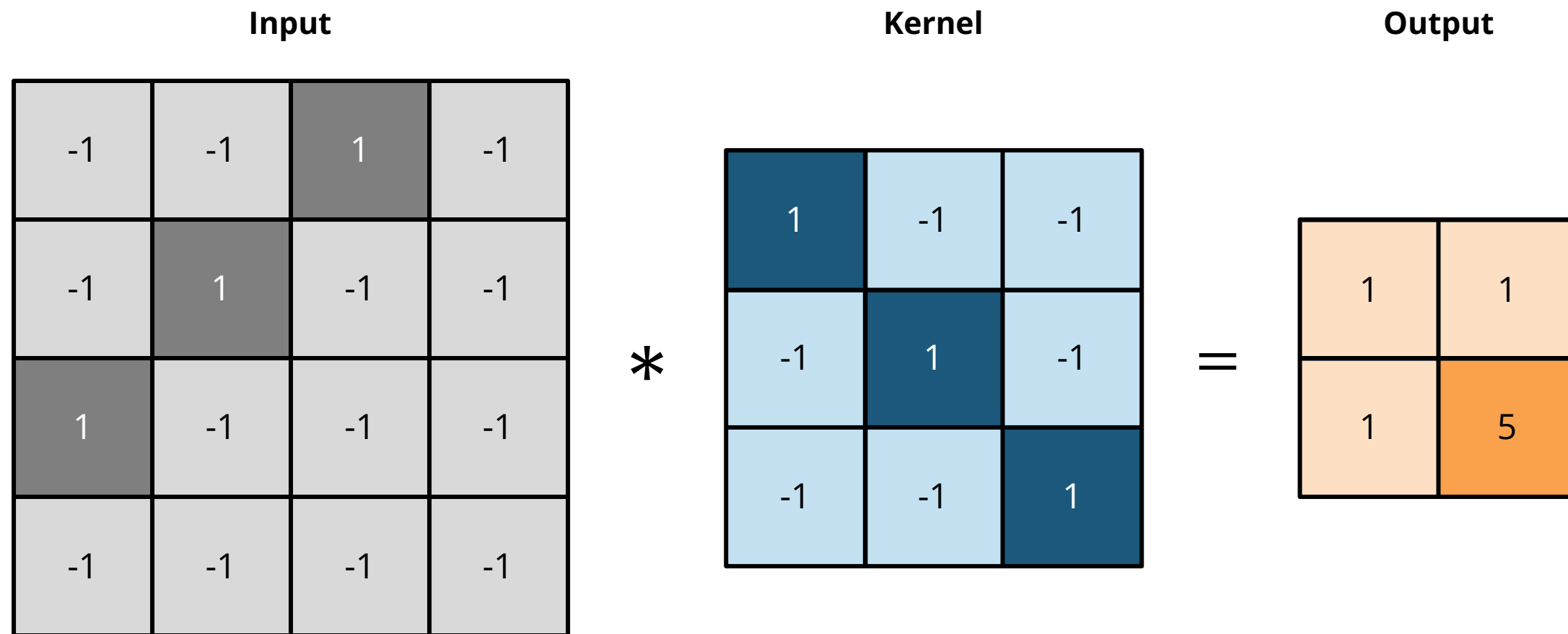
1	-1	-1
-1	1	-1
-1	-1	1

=

Output

9	-1
-1	9

2D Convolution



2D Convolution

Input

-1	-1	1	-1
-1	1	-1	-1
1	-1	-1	-1
-1	-1	-1	-1

*

Kernel

-1	-1	1
-1	1	-1
1	-1	-1

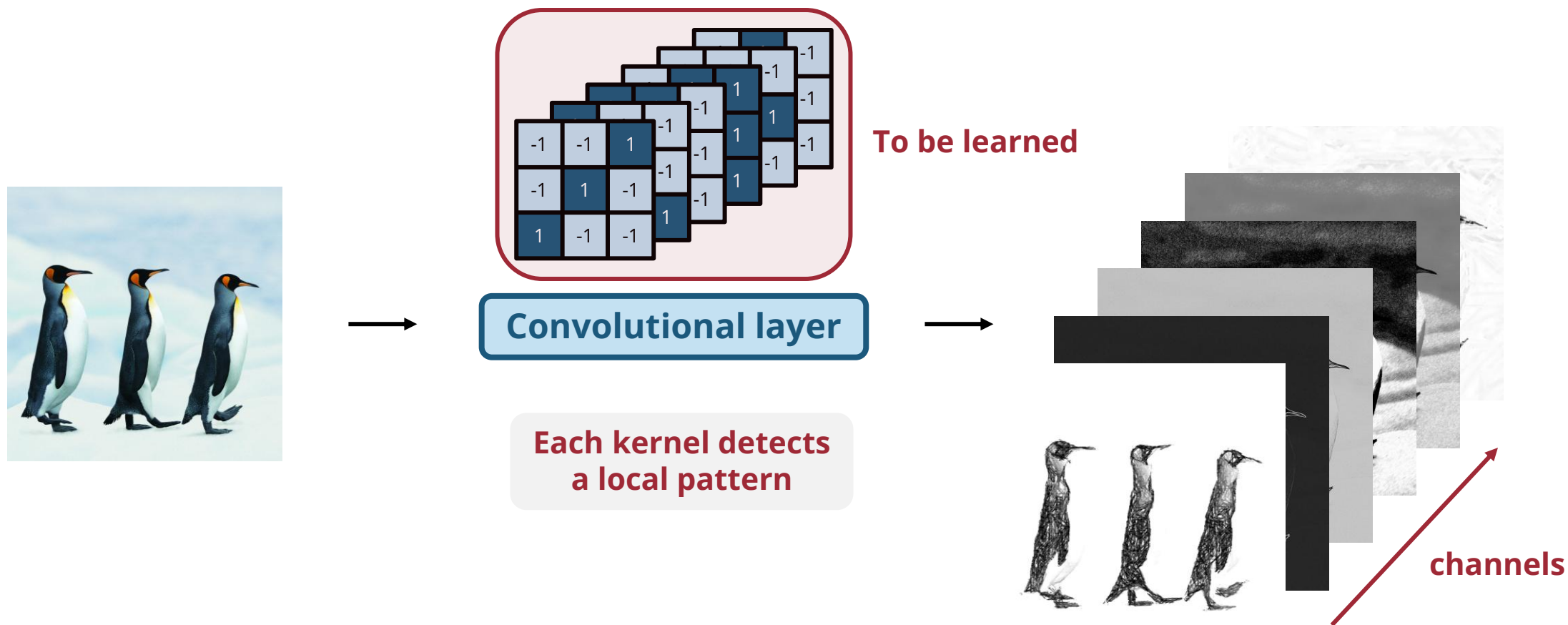
=

Output

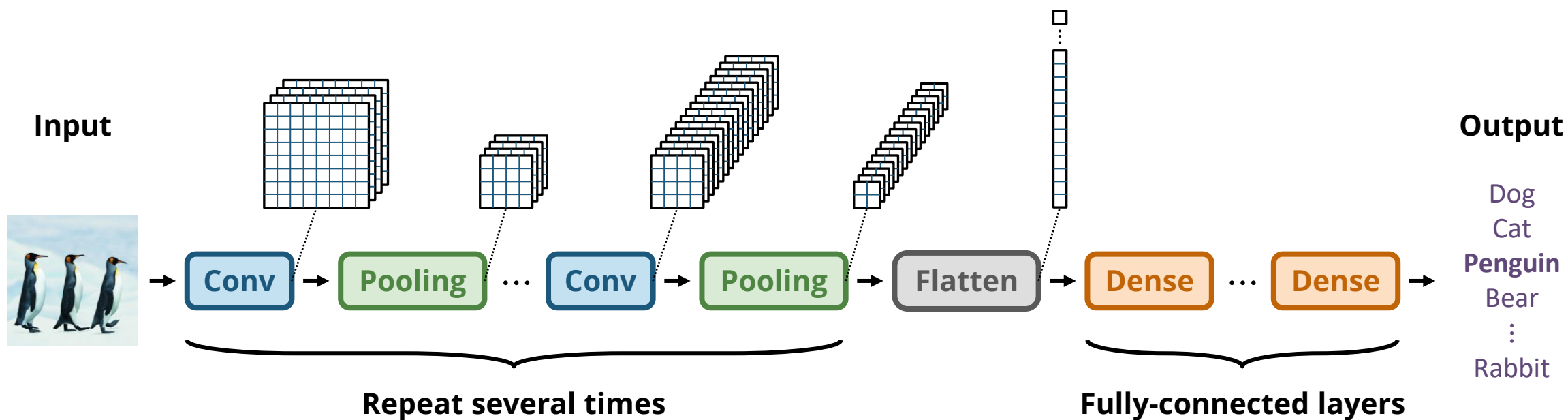
9	-1
-1	1

Convolutional Layer

- A convolutional layer consists of many **learnable kernels** (channels)



Convolutional Neural Network (CNNs)



Padding

padding="valid"

1	-1	-1	-1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

*

1	-1	-1
-1	1	-1
-1	-1	1

=

9	-1
-1	9

padding="same"

0	0	0	0	0	0
0	1	-1	-1	-1	0
0	-1	1	-1	-1	0
0	-1	-1	1	-1	0
0	-1	-1	-1	1	0
0	0	0	0	0	0

*

1	-1	-1
-1	1	-1
-1	-1	1

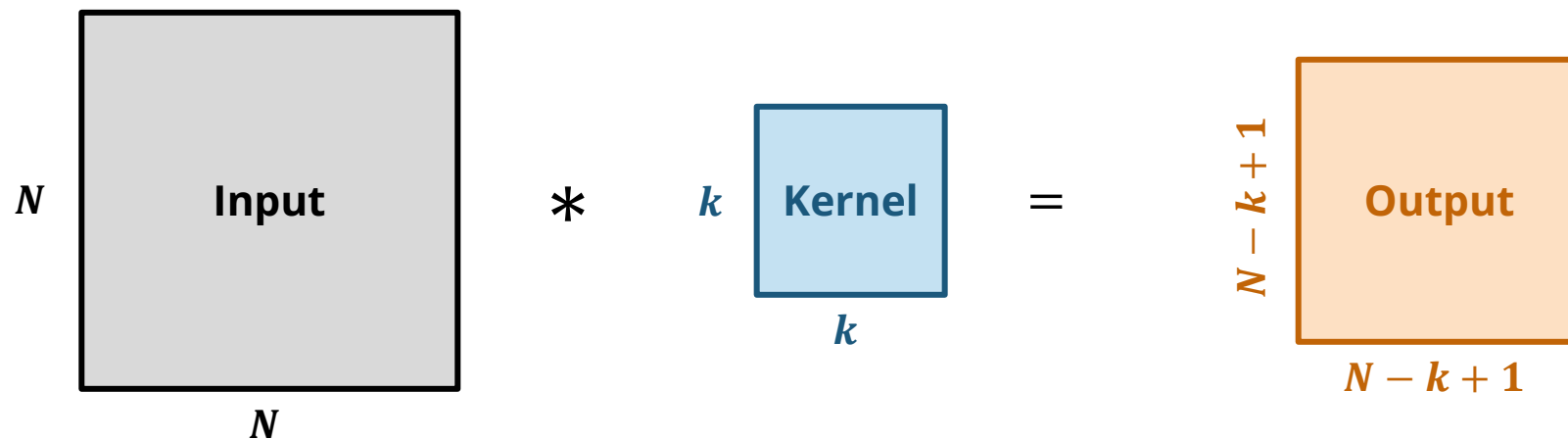
=

4	-2	0	2
-2	9	-1	0
0	-1	9	-2
2	0	-2	4

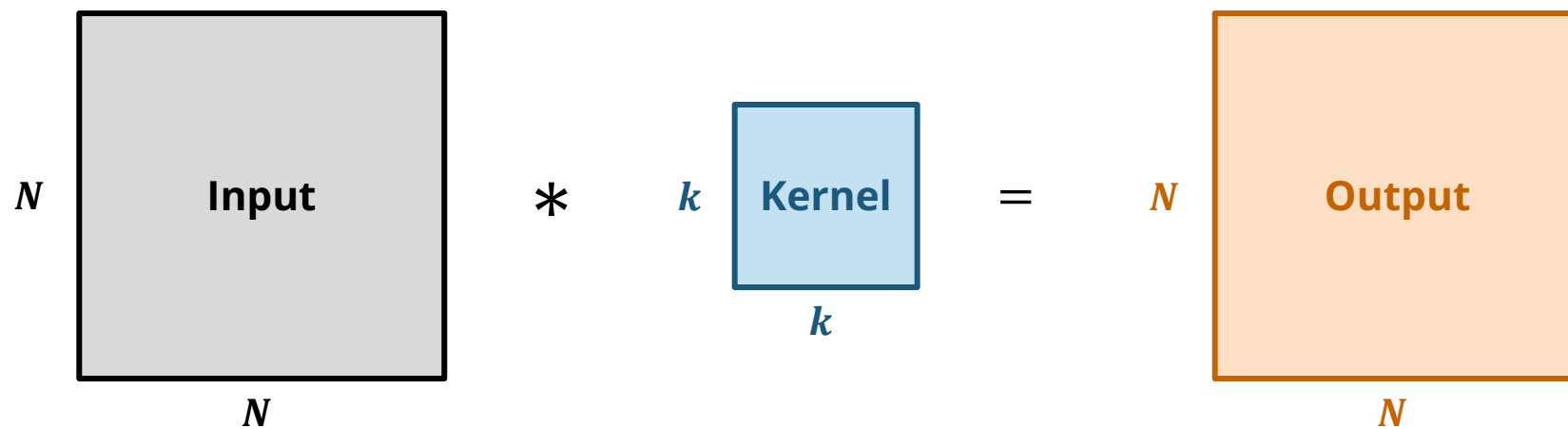
Keep the output of the same size as the input

Shapes

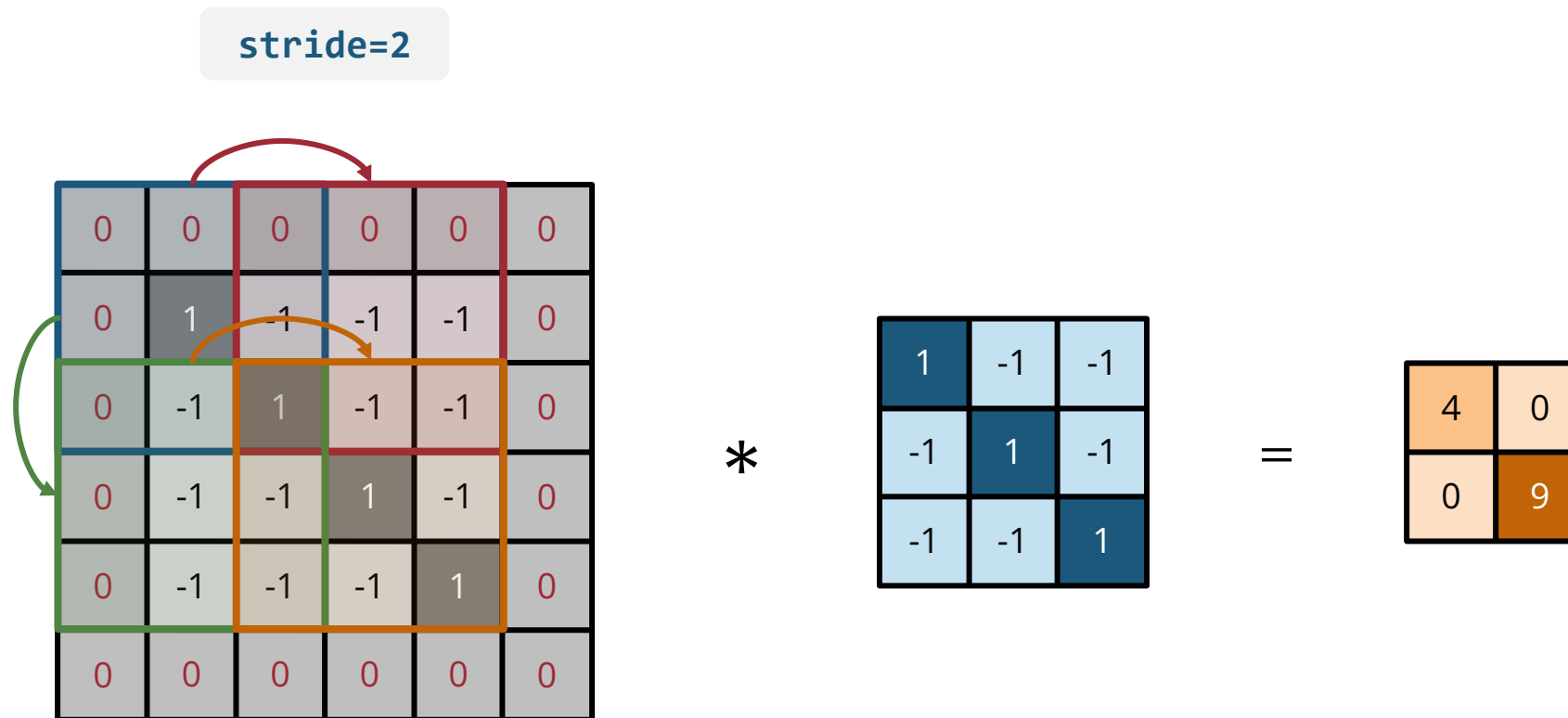
padding="valid"



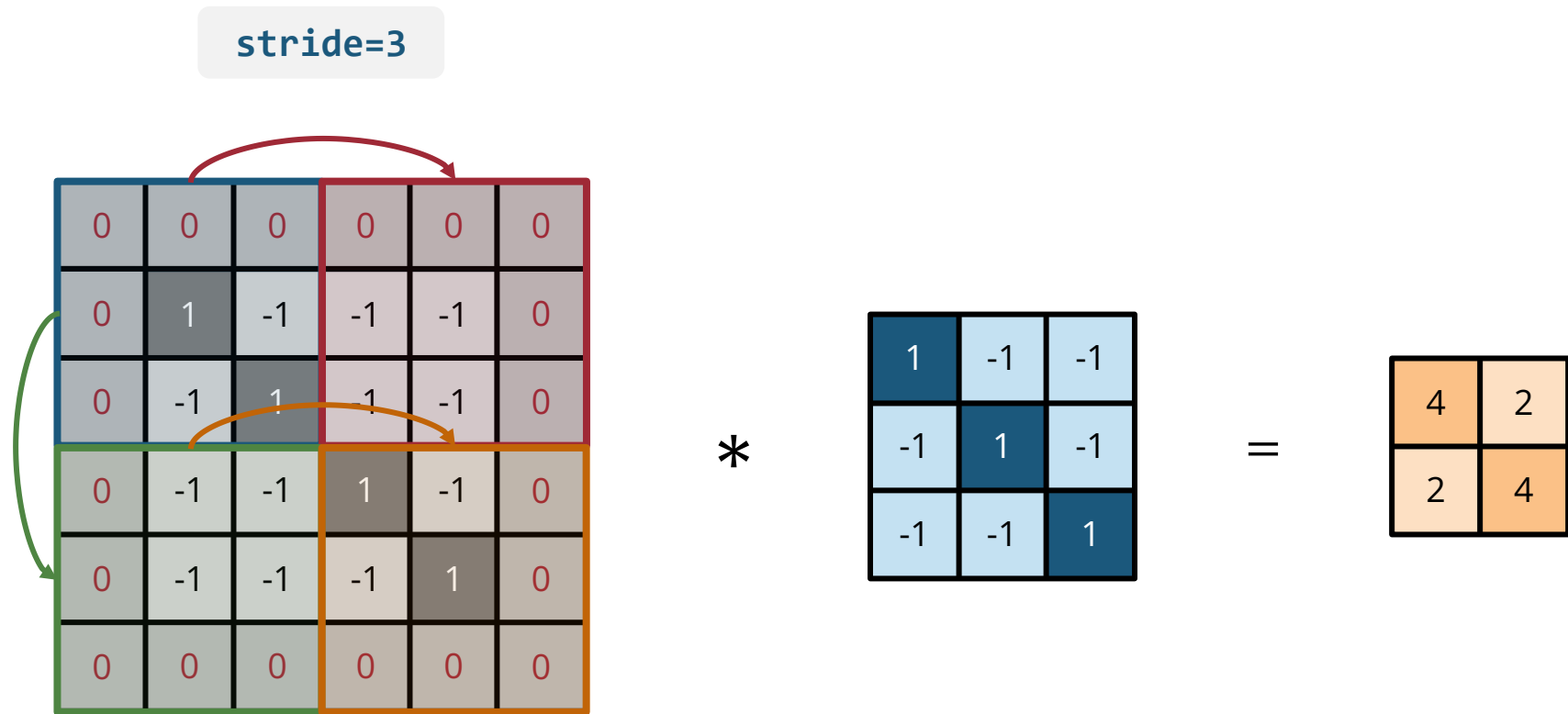
padding="same"



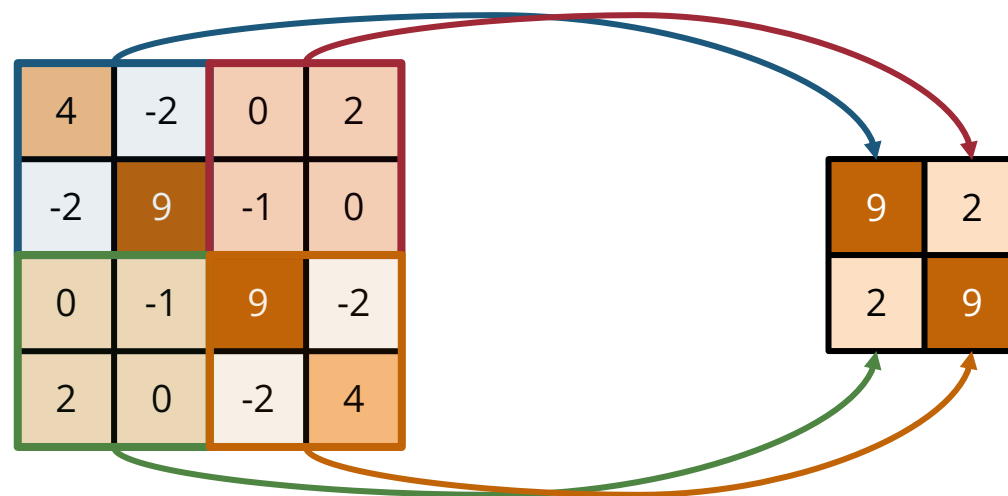
Striding



Striding

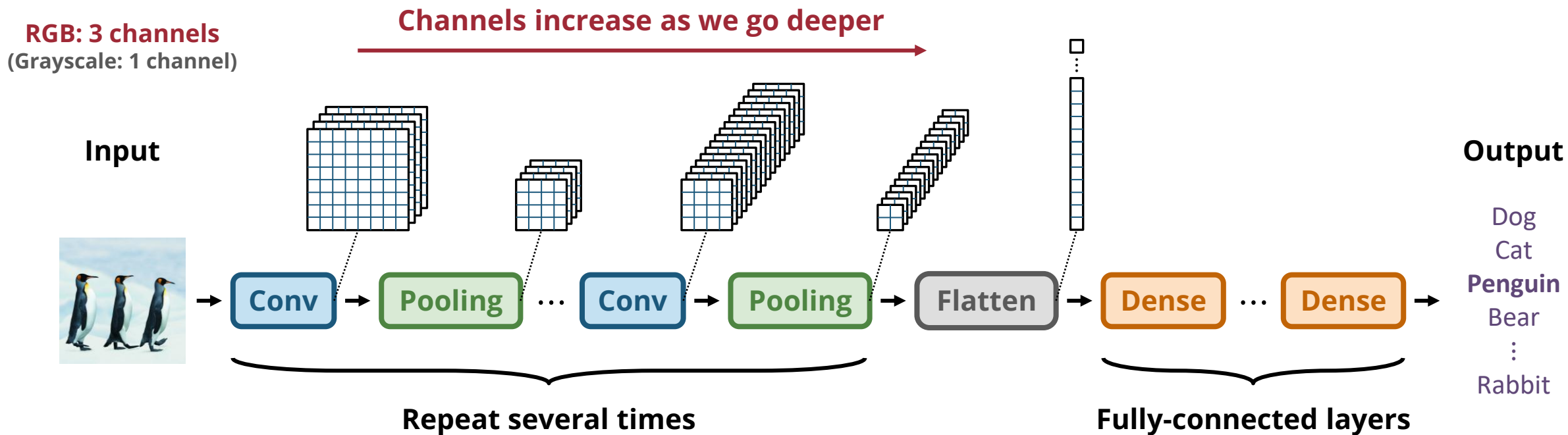


Max Pooling Layer



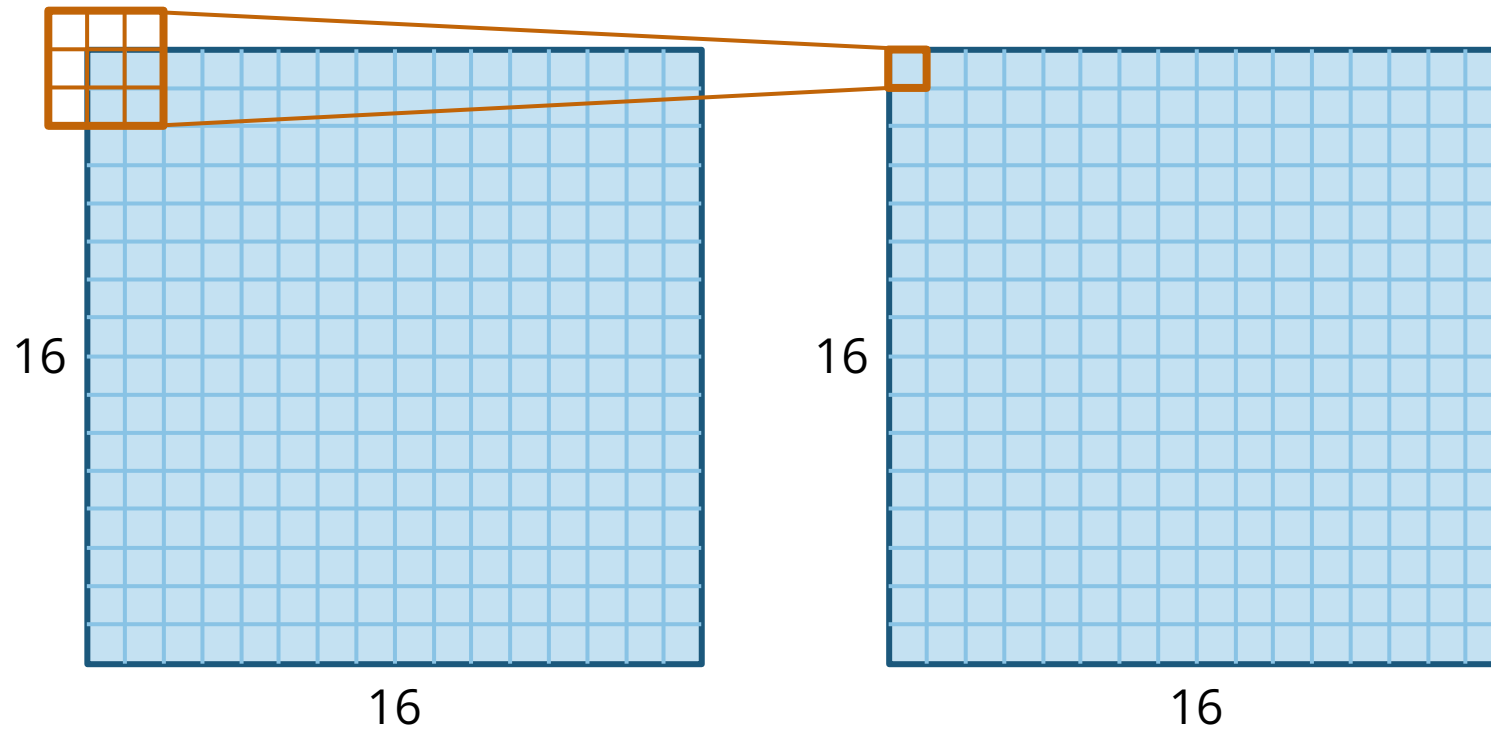
Downsample and keep the strongest activation in each block

Convolutional Neural Network (CNNs)



| A Toy Example

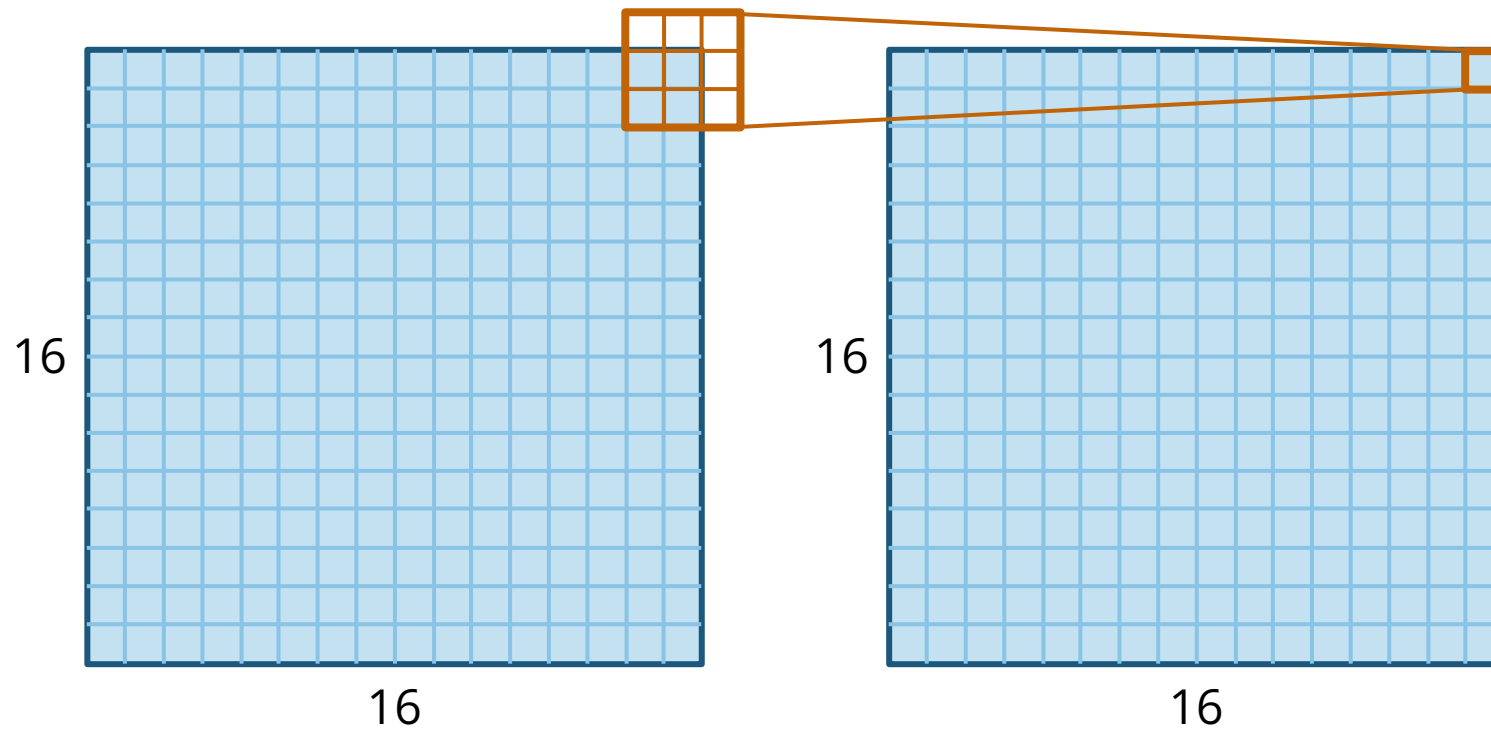
Convolutional layer



```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

| A Toy Example

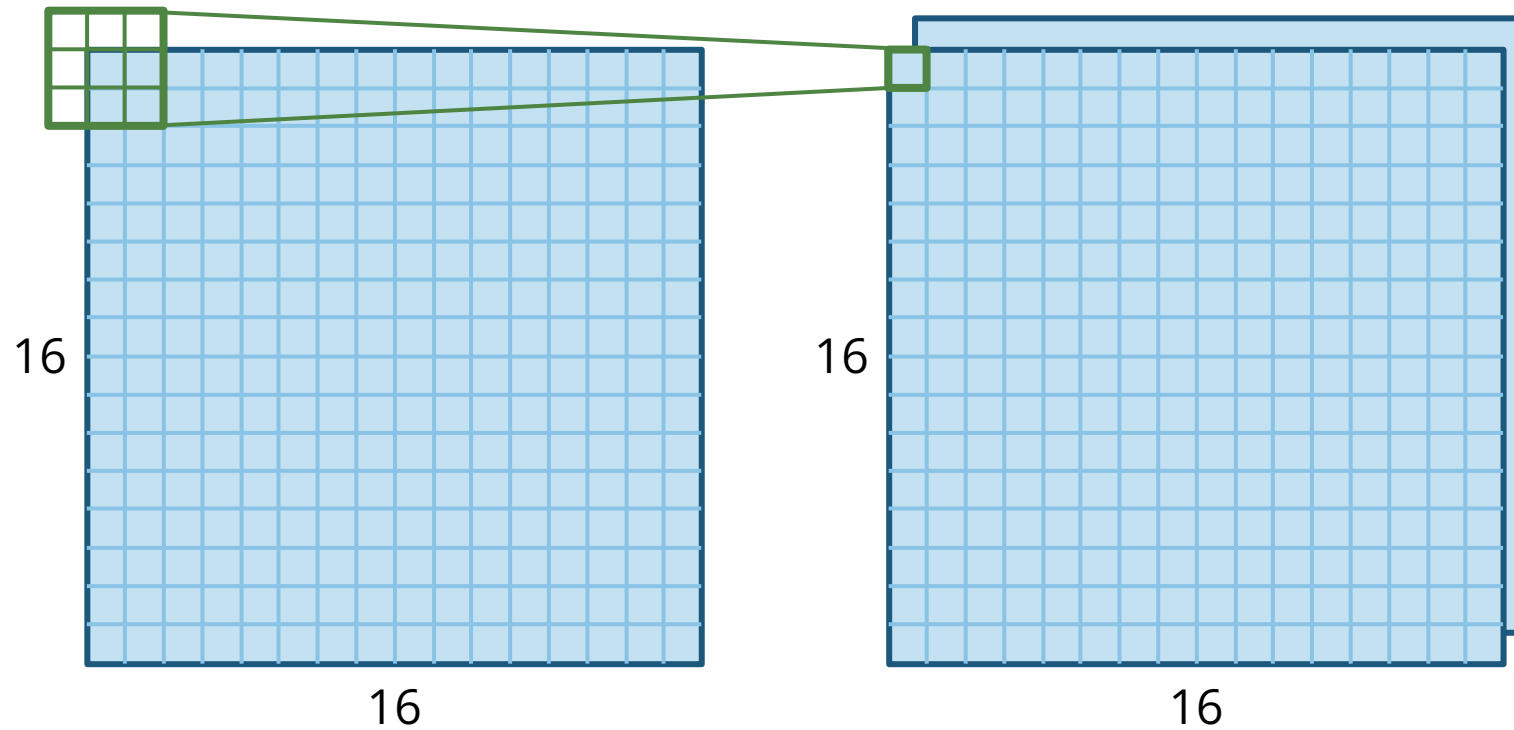
Convolutional layer



```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

A Toy Example

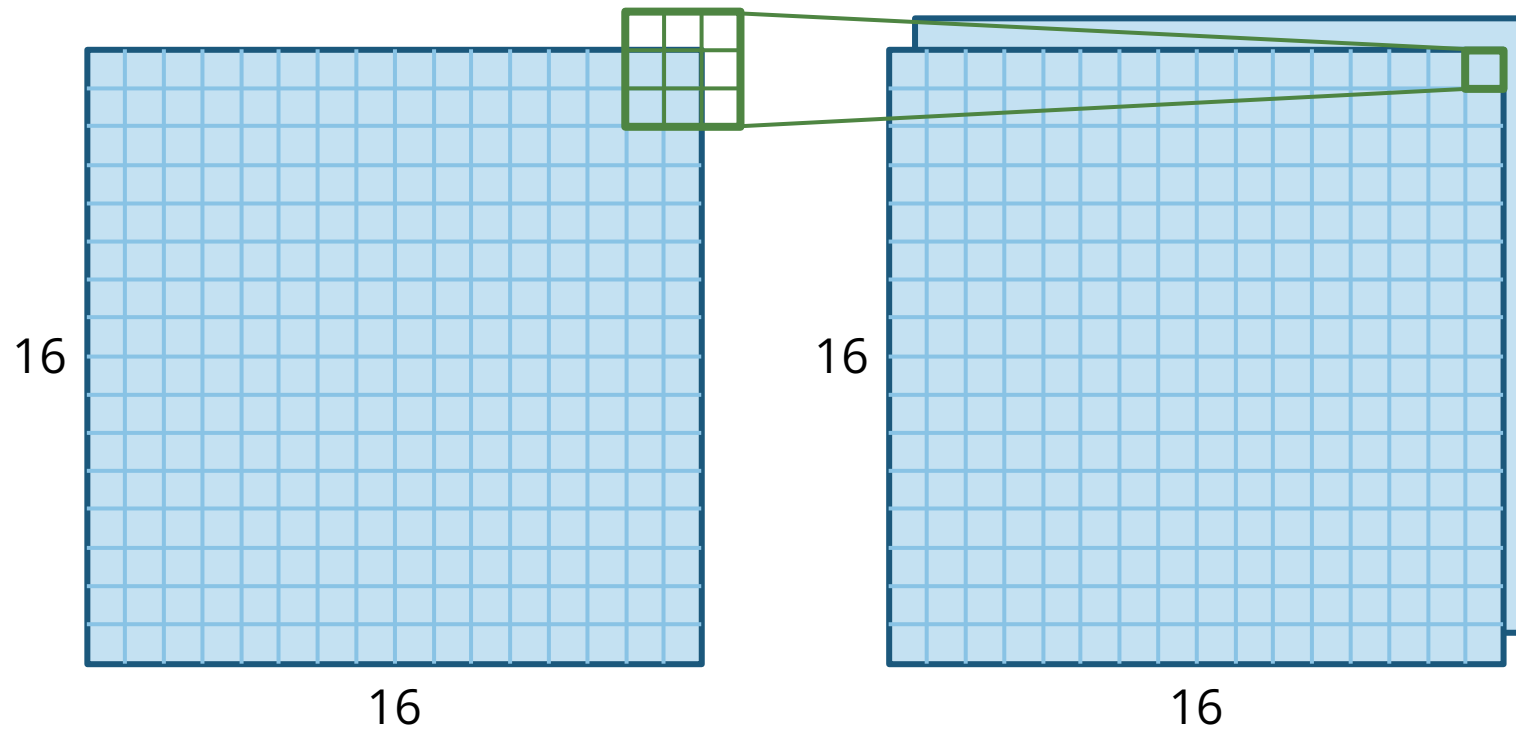
Convolutional layer



```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

| A Toy Example

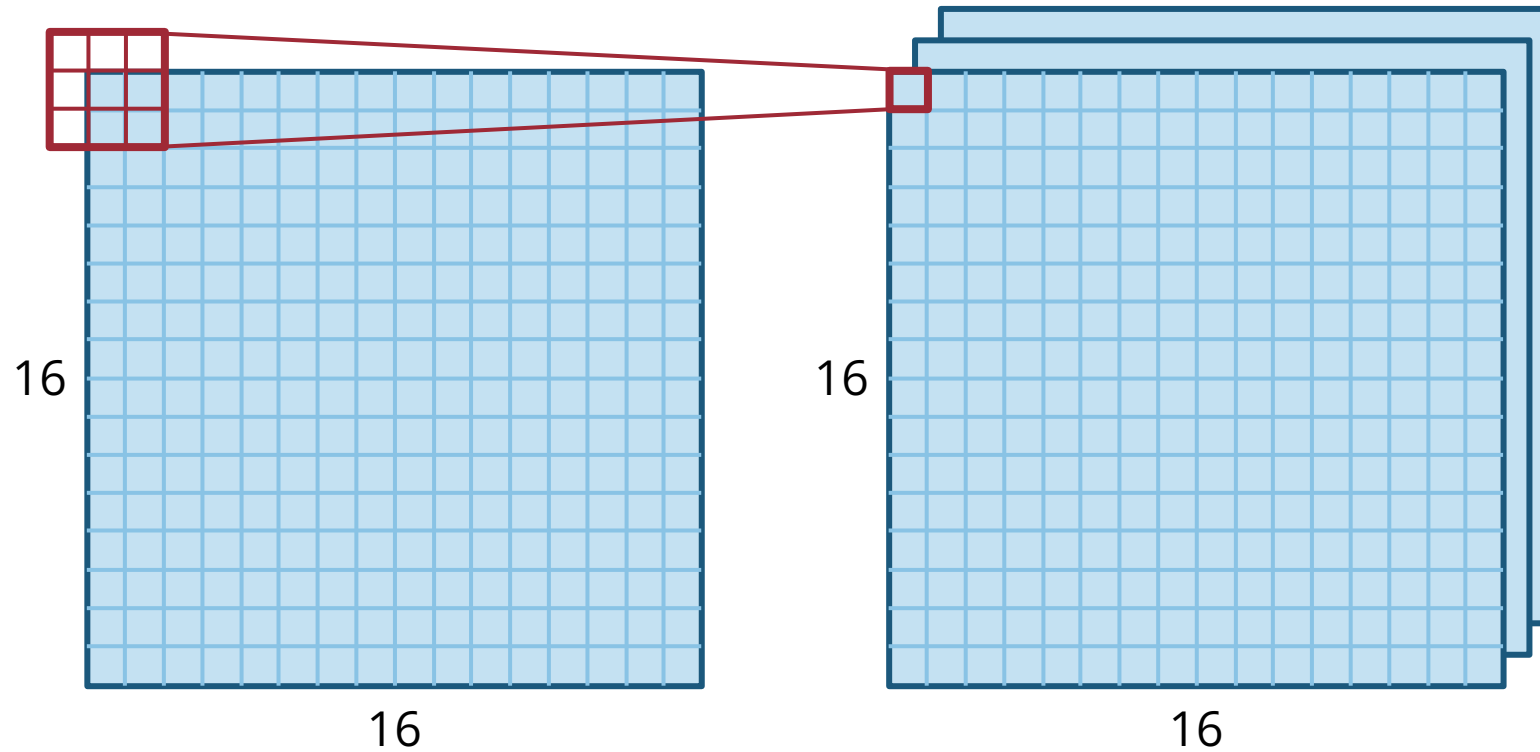
Convolutional layer



```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

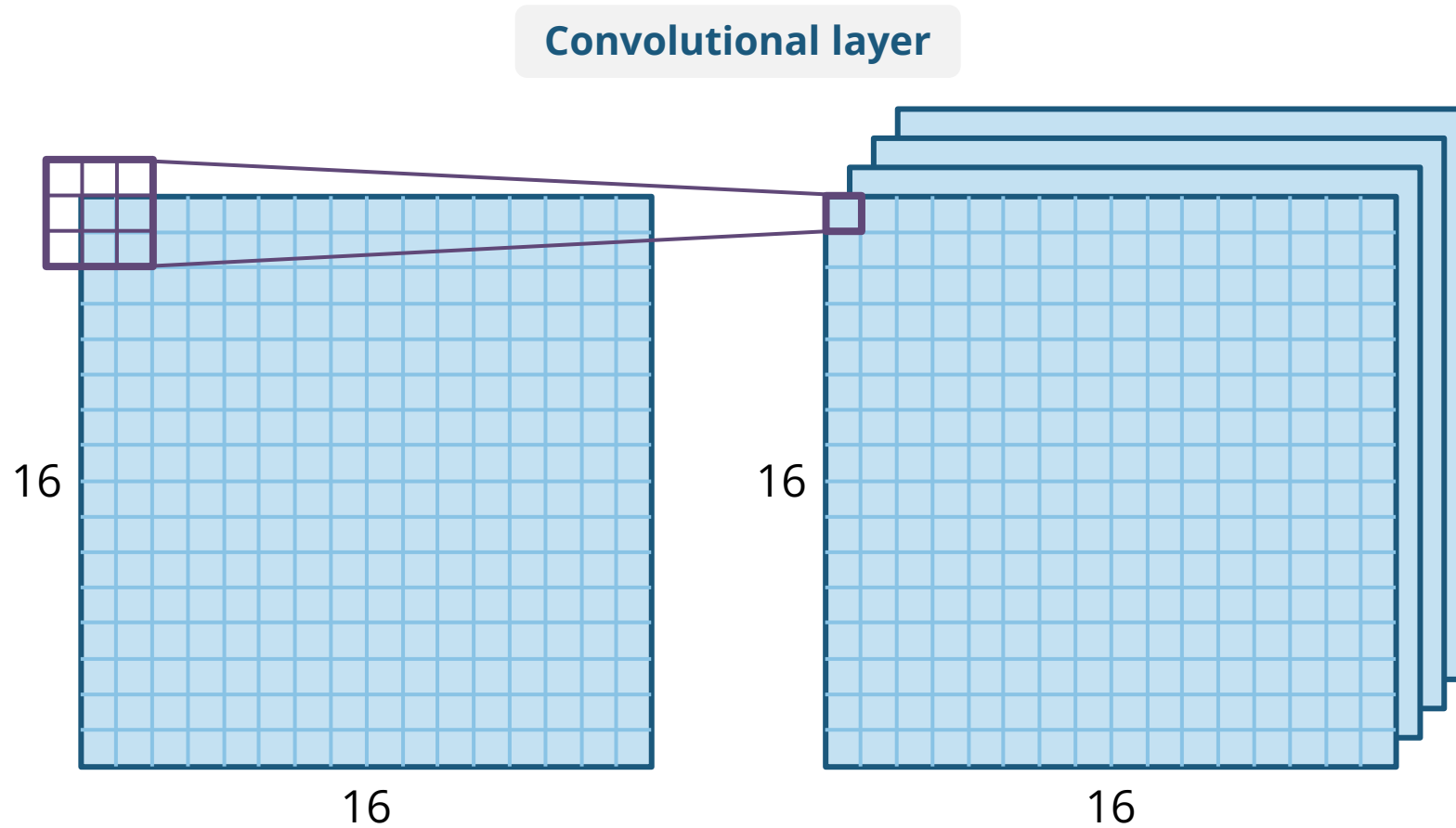
| A Toy Example

Convolutional layer



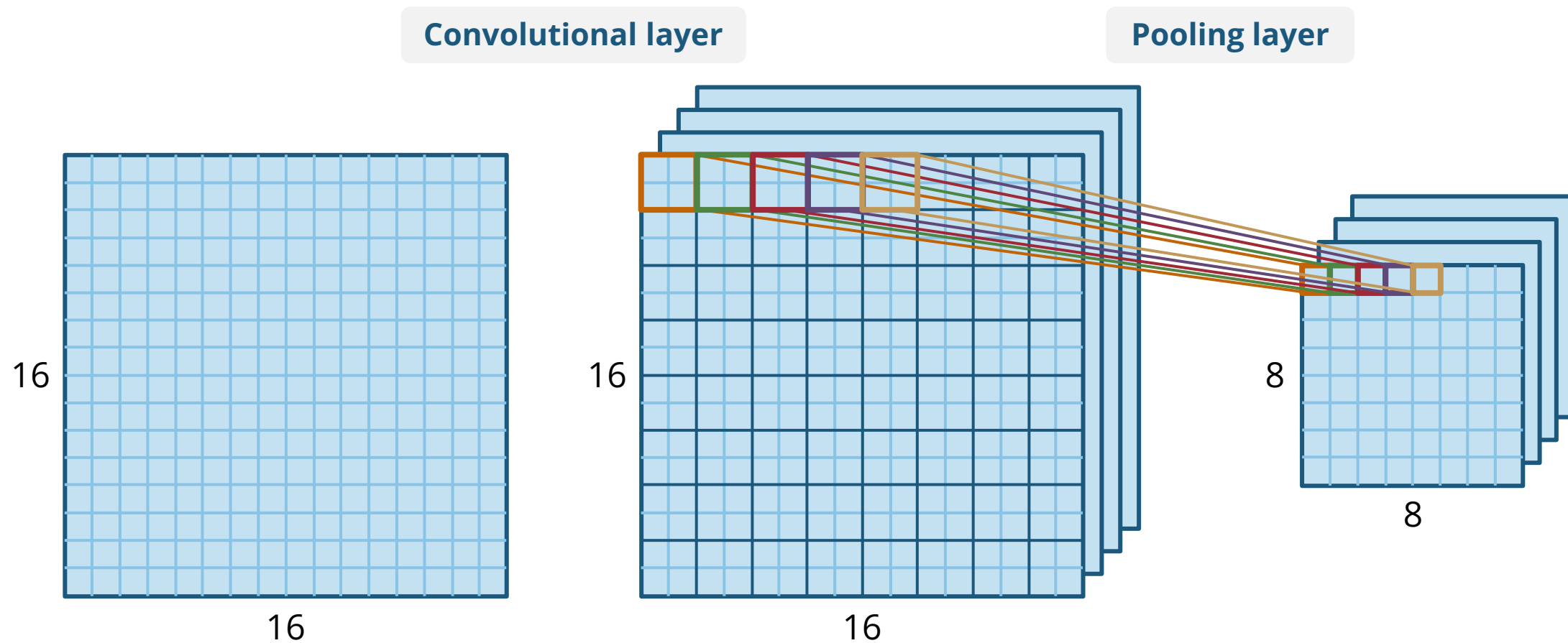
```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

A Toy Example



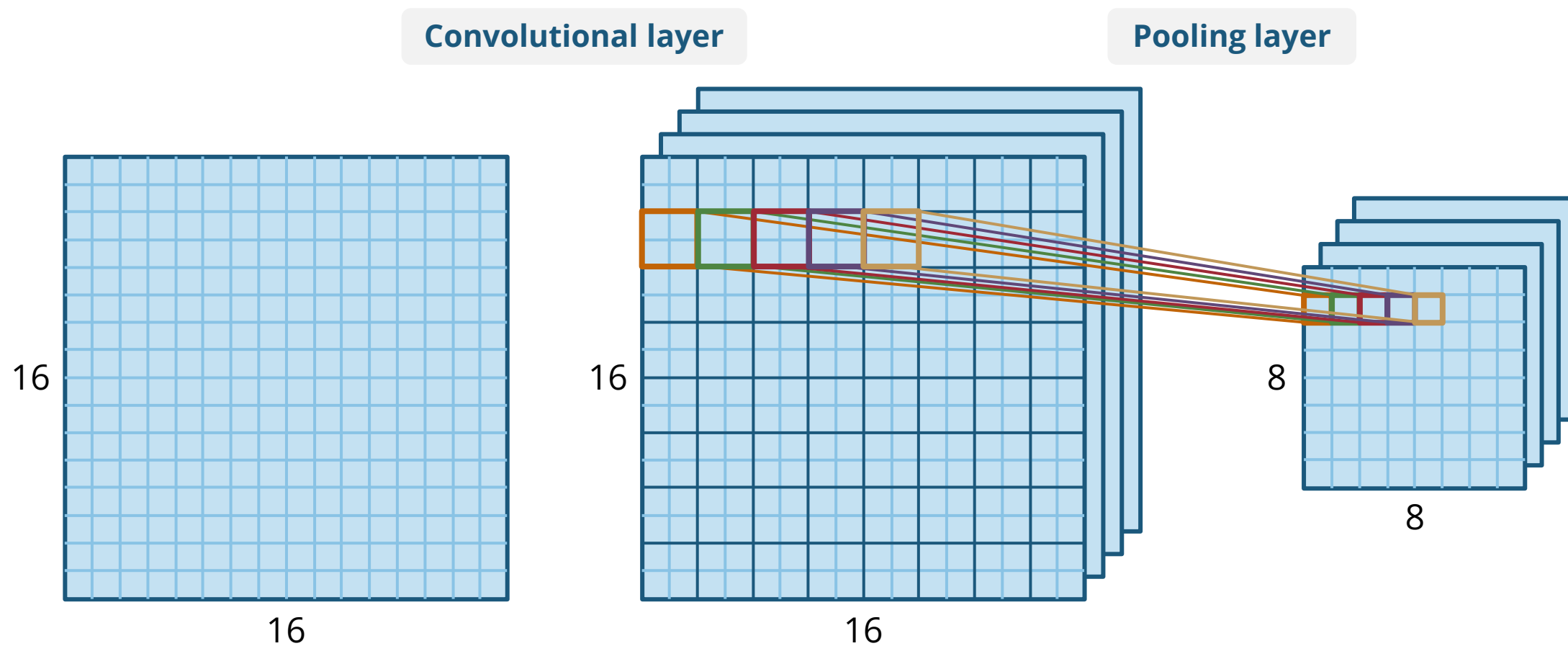
```
nn.Conv2d(in_channels=1, out_channels=4, kernel_size=3, padding="same")
```

A Toy Example



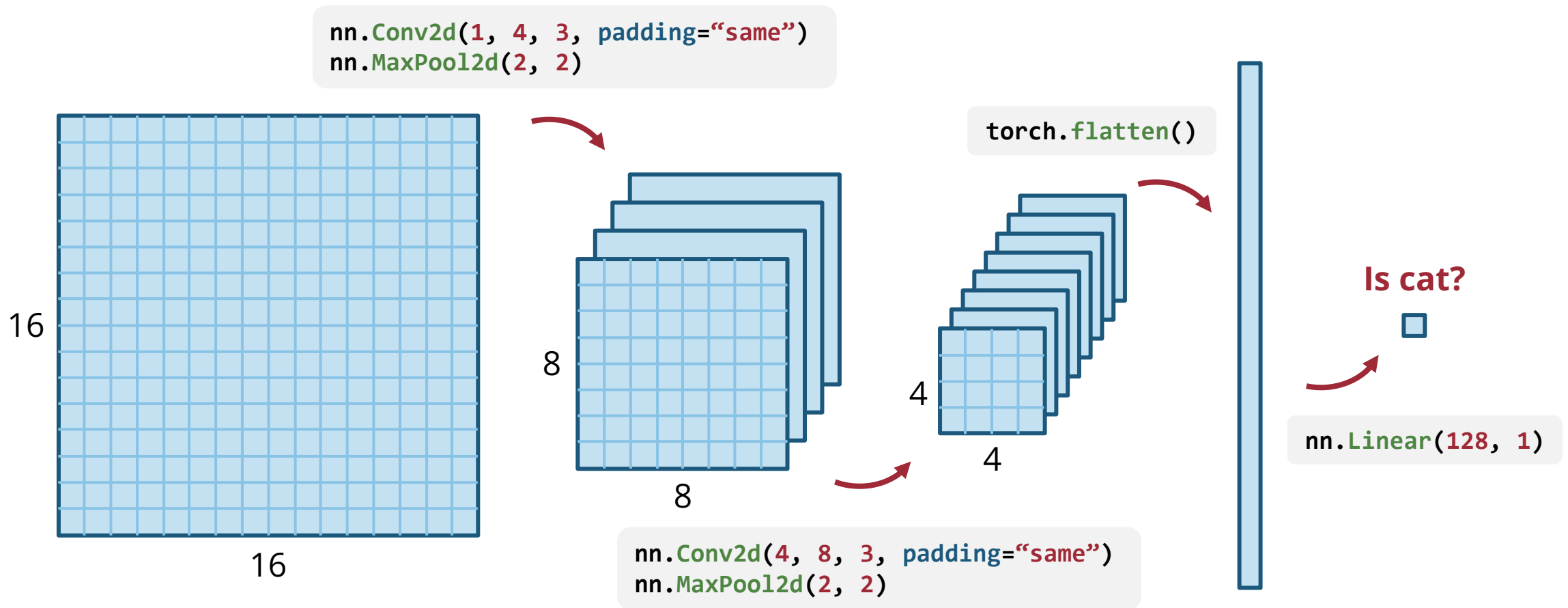
```
nn.MaxPool2d(kernel_size=2, stride=2)
```

A Toy Example

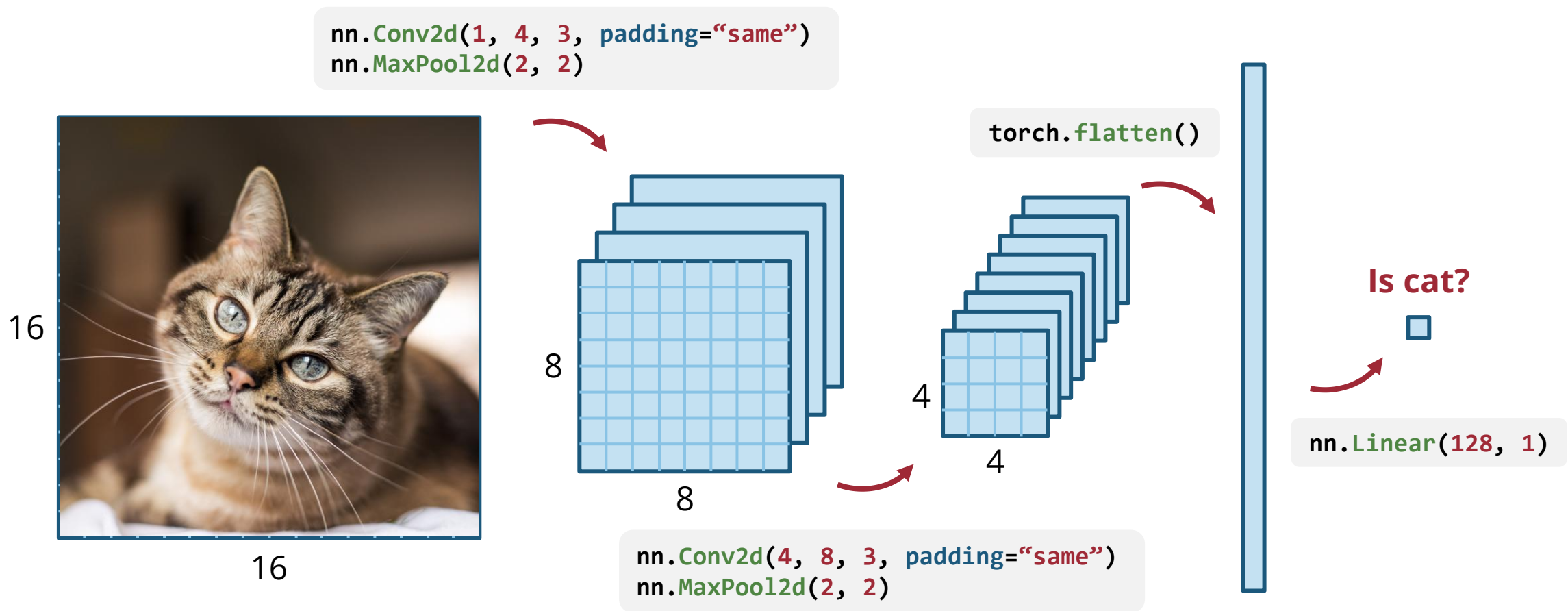


```
nn.MaxPool2d(kernel_size=2, stride=2)
```

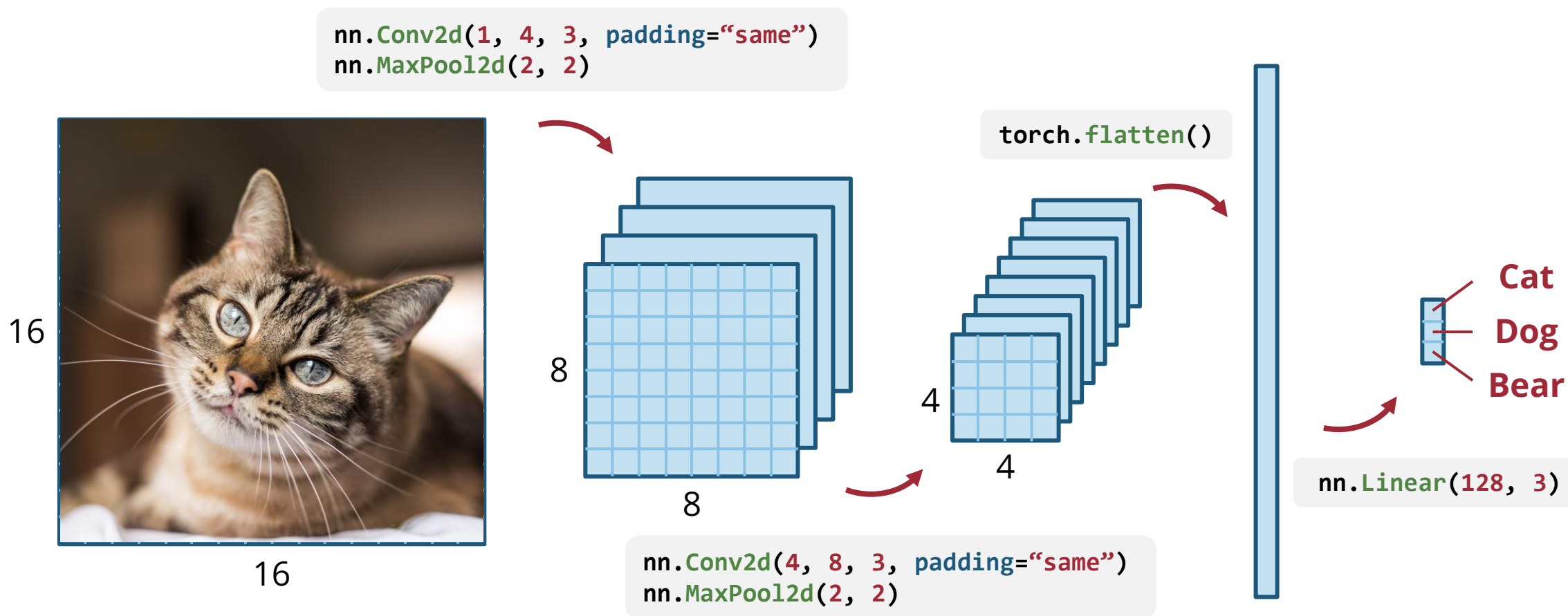
A Toy Example



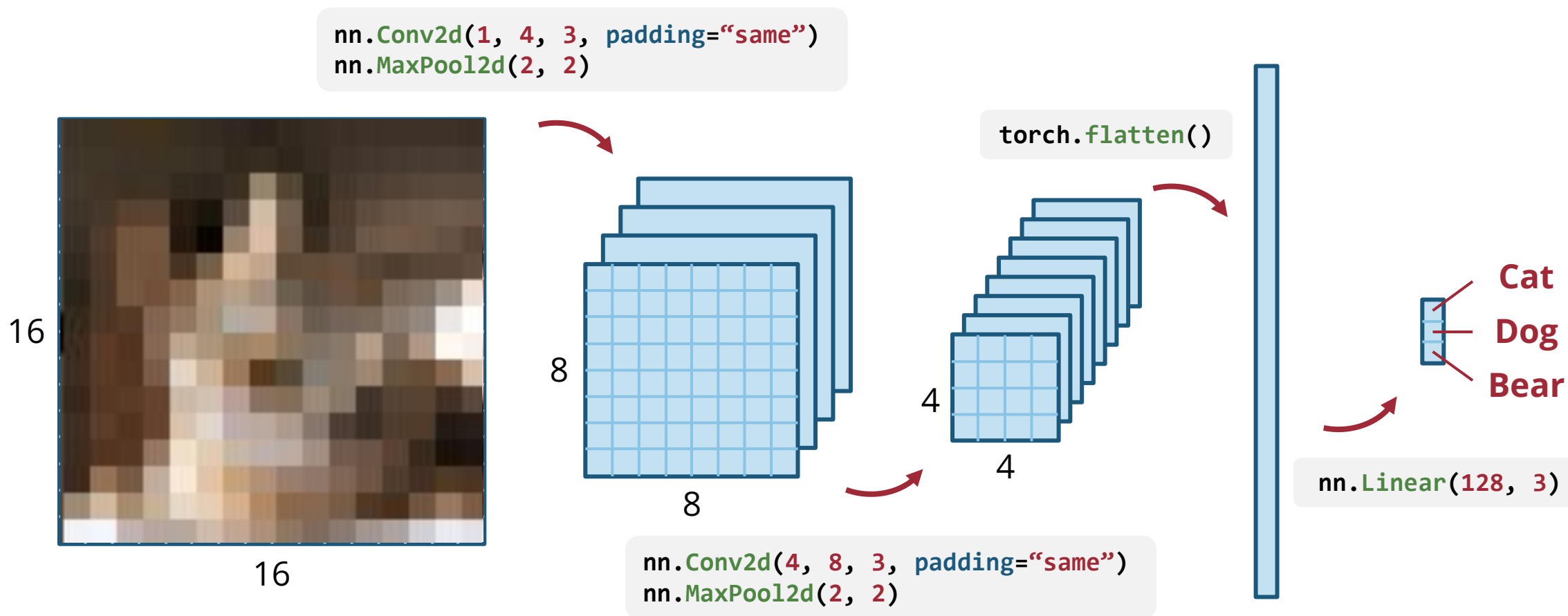
A Toy Example



A Toy Example



A Toy Example



A Real Example

```
class CNN(nn.Module):
    """A basic convolutional neural network."""
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 16, 3, padding="same")
        self.conv2 = nn.Conv2d(16, 32, 3, padding="same")
        self.conv3 = nn.Conv2d(32, 64, 3, padding="same")
        self.conv4 = nn.Conv2d(64, 128, 3, padding="same")
        self.pool = nn.MaxPool2d(2, 2)
        self.fc = nn.Linear(128 * 4 * 4, n_classes)

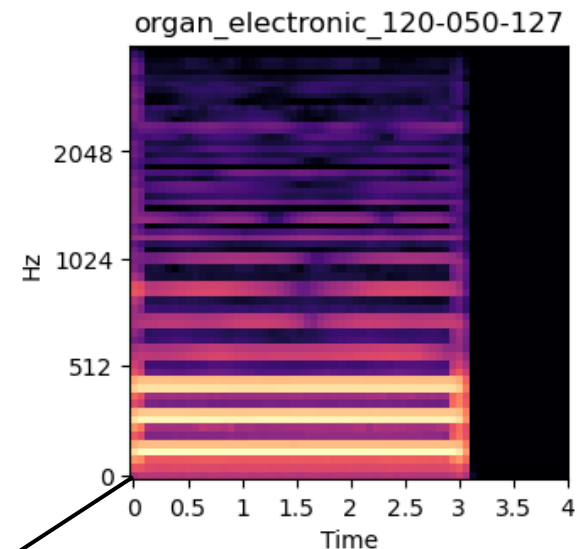
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        x = torch.flatten(x, 1)
        x = self.fc(x)
        return x
```

```
def forward(self, x):
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = self.pool(F.relu(self.conv3(x)))
    x = self.pool(F.relu(self.conv4(x)))
    x = torch.flatten(x, 1)
    x = self.fc(x)
    return x
```

Input: 1 x 64 x 64

16	x	32 x 32	=	16384
32	x	16 x 16	=	8192
64	x	8 x 8	=	4096
128	x	4 x 4	=	2048

More channels,
lower resolution
as we go deeper



Total number of
features decrease
as we go deeper

A Real Example

```
class CNN(nn.Module):  
    """A basic convolutional neural network."""  
    def __init__(self):  
        super().__init__()  
        self.conv1 = nn.Conv2d(1, 16, 3, padding="same")  
        self.conv2 = nn.Conv2d(16, 32, 3, padding="same")  
        self.conv3 = nn.Conv2d(32, 64, 3, padding="same")  
        self.conv4 = nn.Conv2d(64, 128, 3, padding="same")  
        self.pool = nn.MaxPool2d(2, 2)  
        self.fc = nn.Linear(128 * 4 * 4, n_classes)  
  
    def forward(self, x):  
        x = self.pool(F.relu(self.conv1(x)))  
        x = self.pool(F.relu(self.conv2(x)))  
        x = self.pool(F.relu(self.conv3(x)))  
        x = self.pool(F.relu(self.conv4(x)))  
        x = torch.flatten(x, 1)  
        x = self.fc(x)  
        return x
```

Input channels (points to 1 in conv1)
Output channels (points to 16 in conv1)
Kernel size (points to 3 in conv1)

How many parameters do we have in each layer?

$(3 \times 3 \times 1 + 1) \times 16$	= 160
$(3 \times 3 \times 16 + 1) \times 32$	= 4640
$(3 \times 3 \times 32 + 1) \times 64$	= 18496
$(3 \times 3 \times 64 + 1) \times 128$	= 73856
$(2048 + 1) \times 11$	= 22539

| Benefits of CNNs

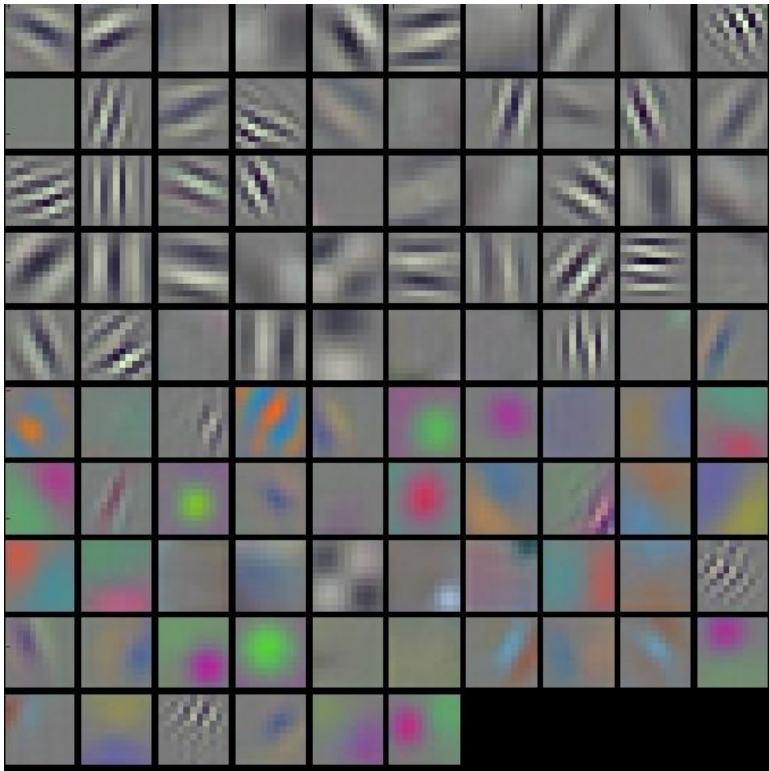
- Learn **local patterns**
- **Invariant to shifts**
 - Also called *translational invariance*
- **Reuse the learned filters** across
 - Different parts of the image
 - Across different images
- **Higher parameter-efficiency** against fully-connected neural network

What does a CNN Learn?

Learned CNN Kernels in a Trained AlexNet

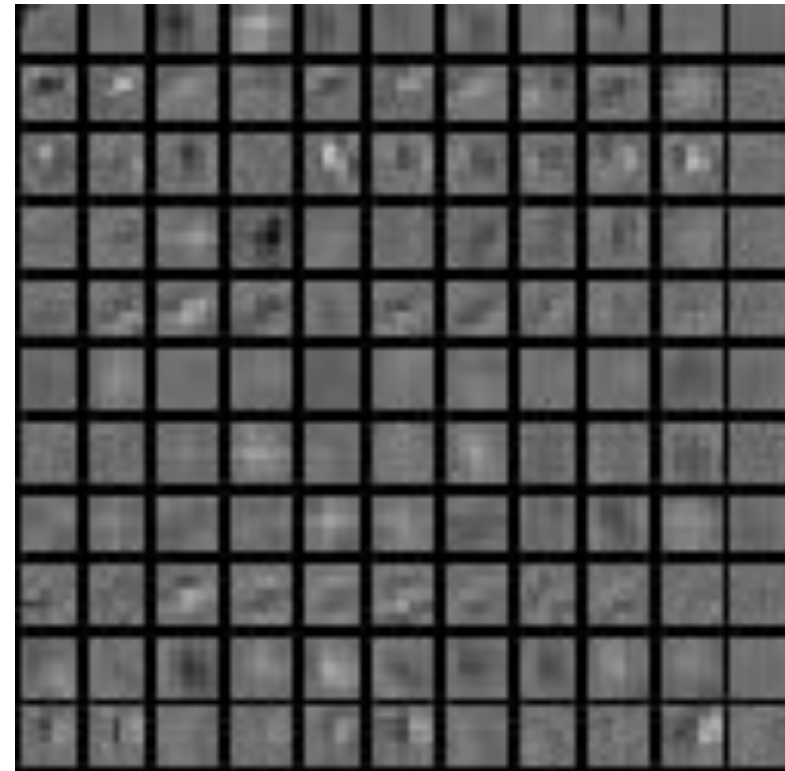
1st convolutional layer

11x11
kernels

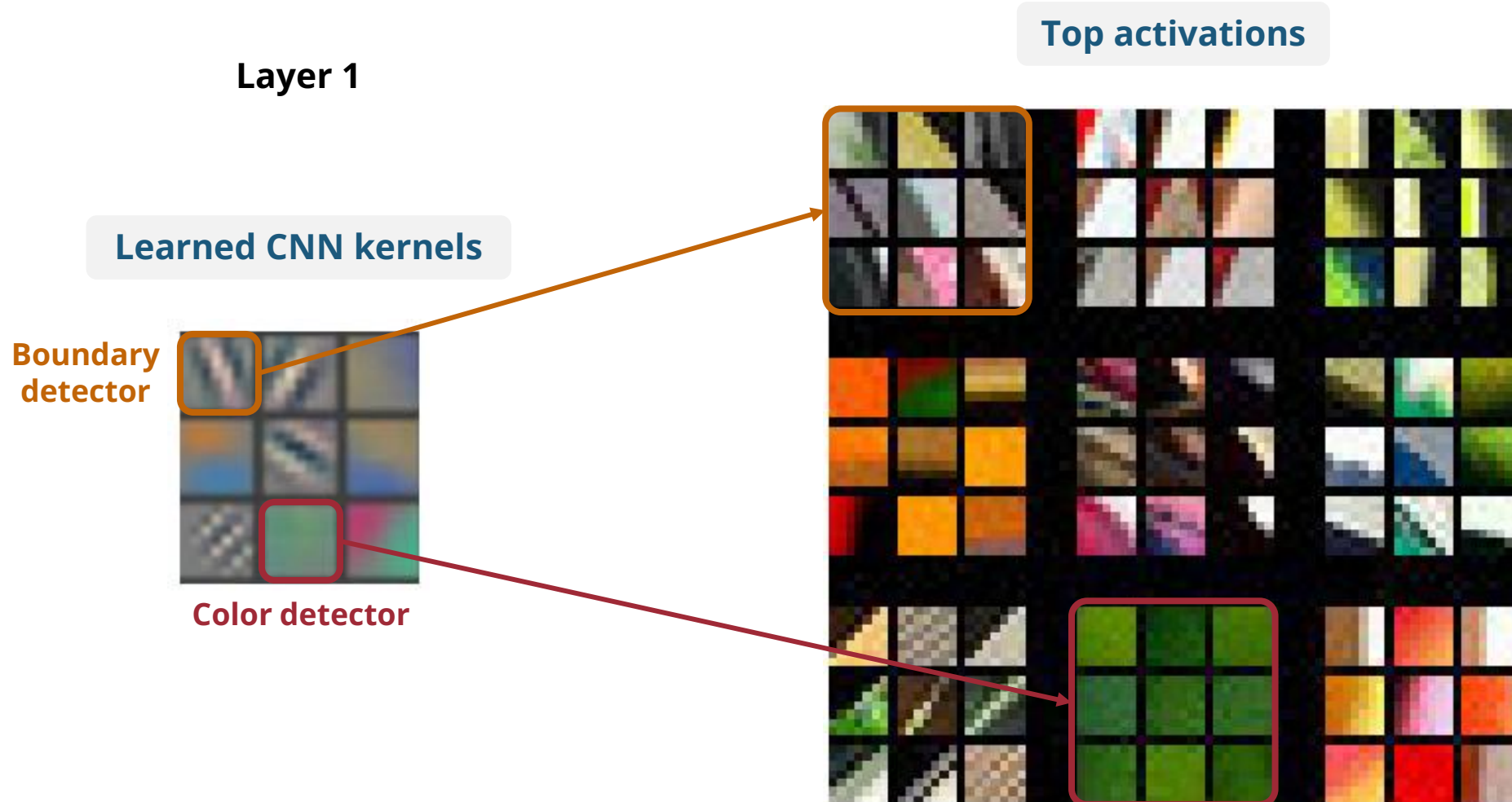


2nd convolutional layer

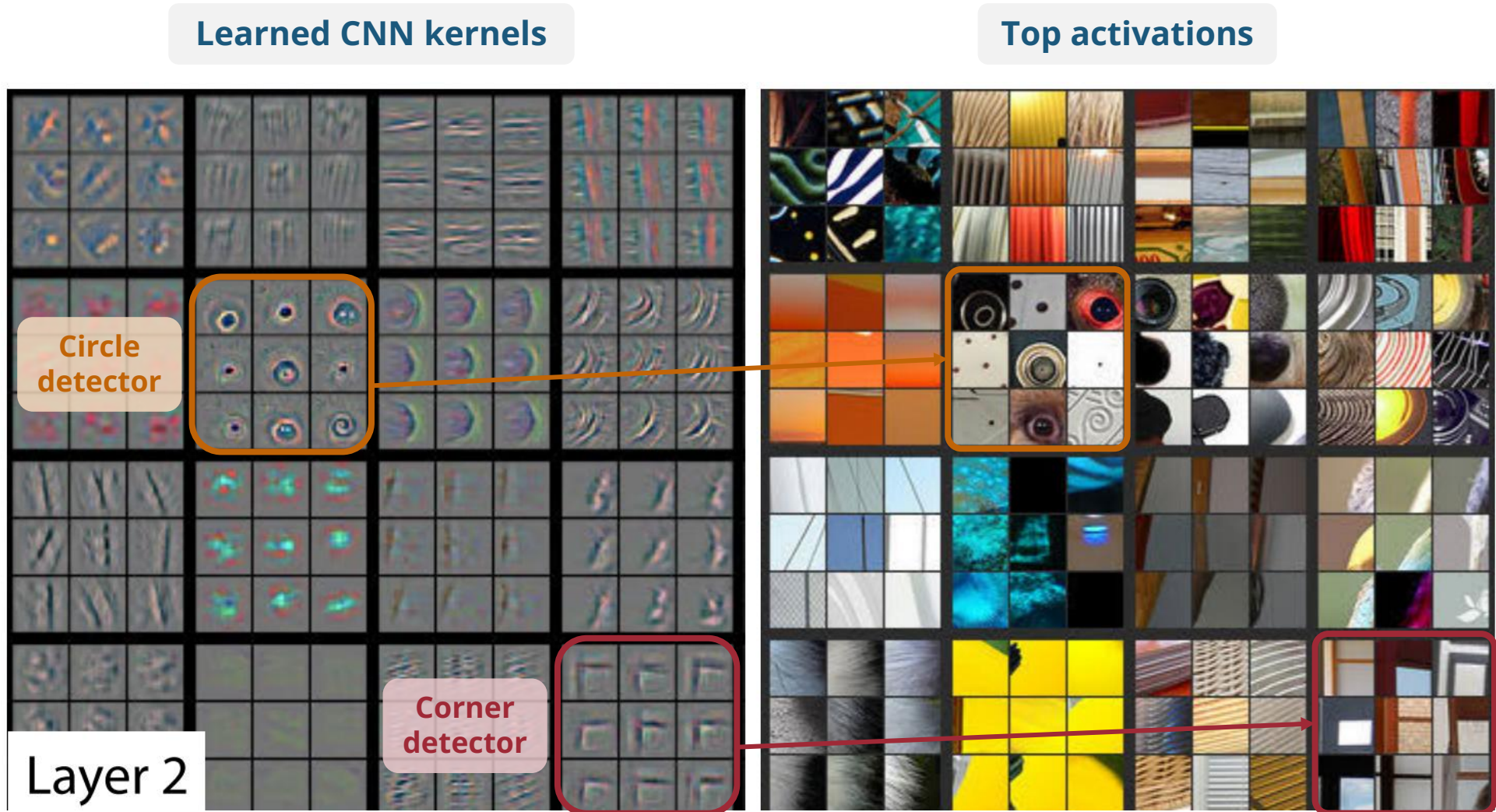
5x5
kernels



Learned CNN Kernels in a Trained AlexNet



Learned CNN Kernels in a Trained AlexNet

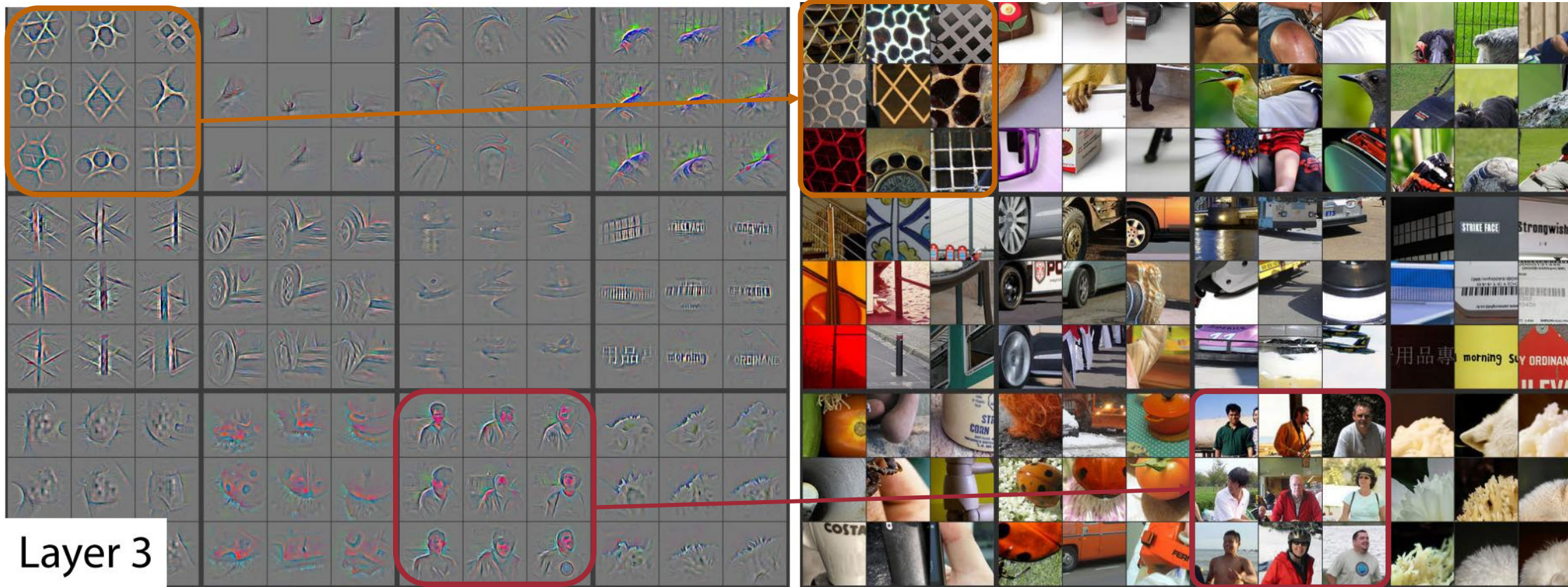


Learned CNN Kernels in a Trained AlexNet

Learned CNN kernels

Top activations

Grid detector

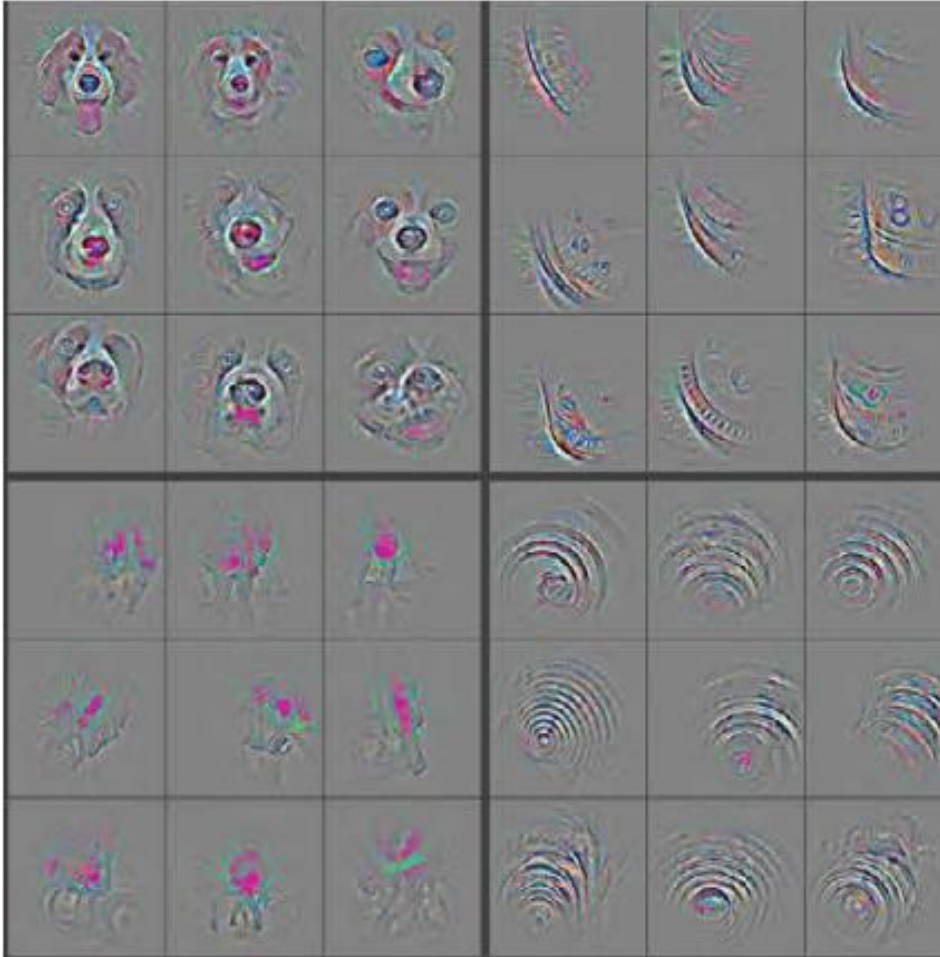


Layer 3

Human detector

Learned CNN Kernels in a Trained AlexNet

Learned CNN kernels

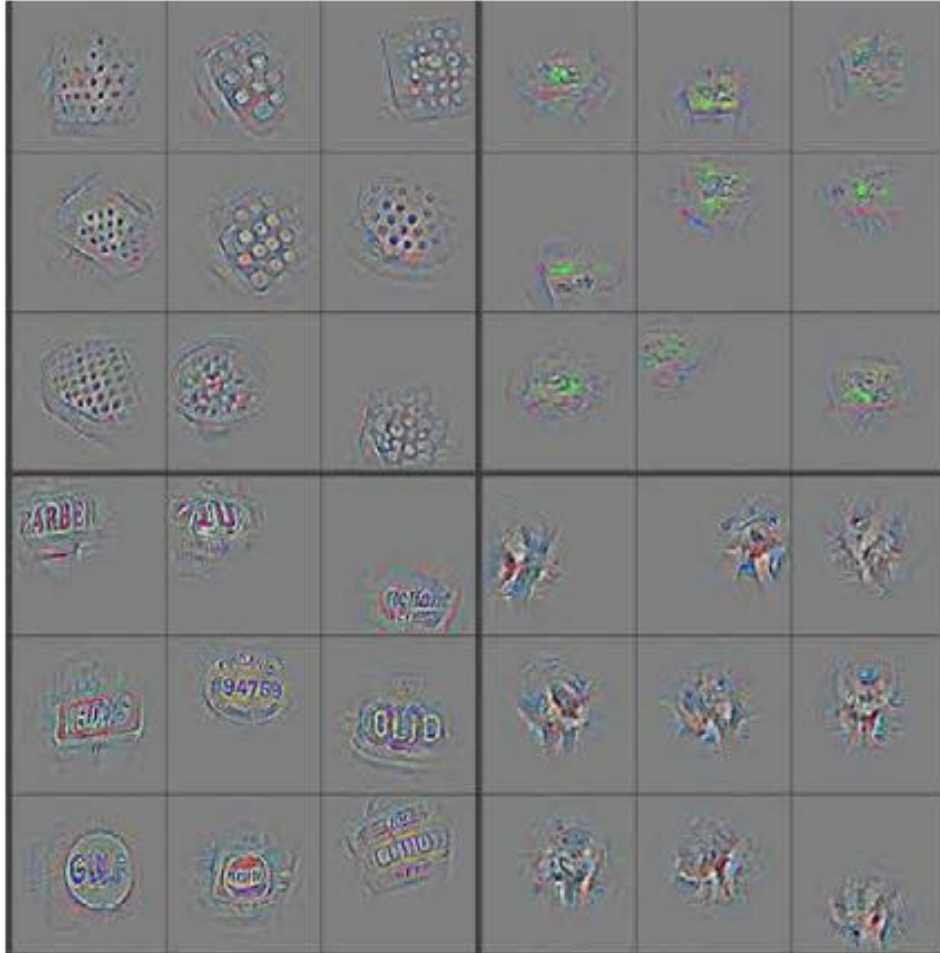


Top activations



Learned CNN Kernels in a Trained AlexNet

Learned CNN kernels

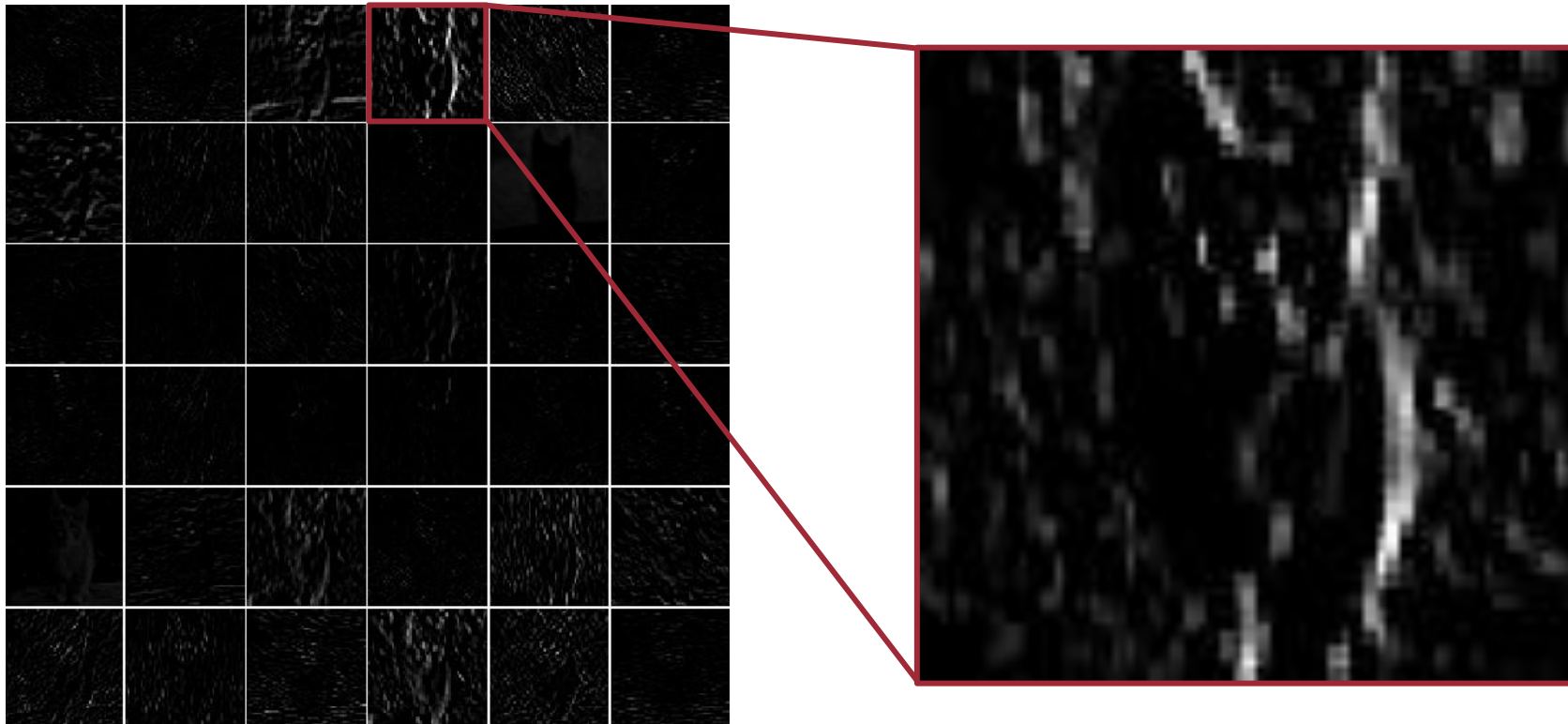


Top activations



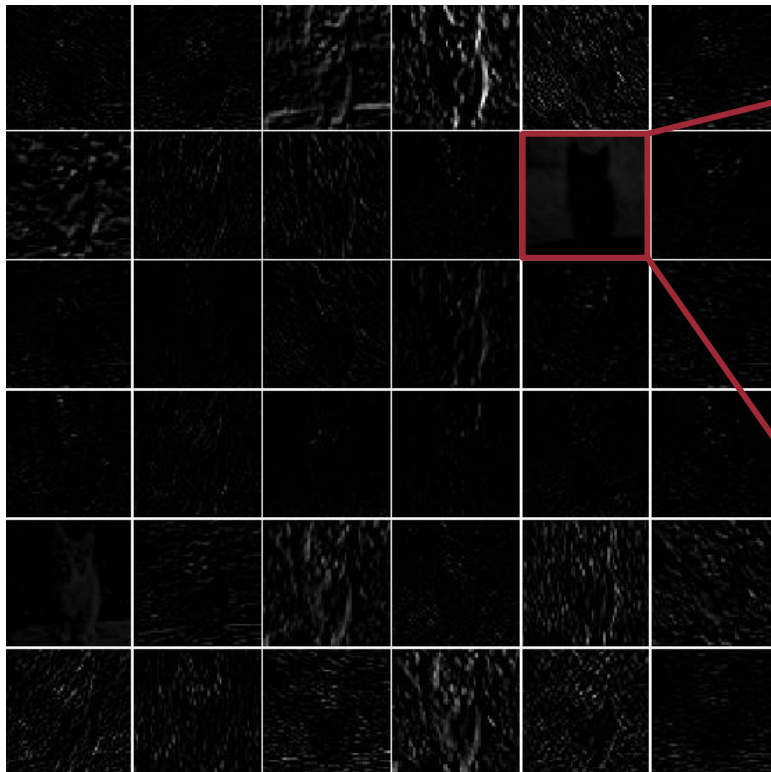
Activations in a Trained AlexNet

1st convolutional layer

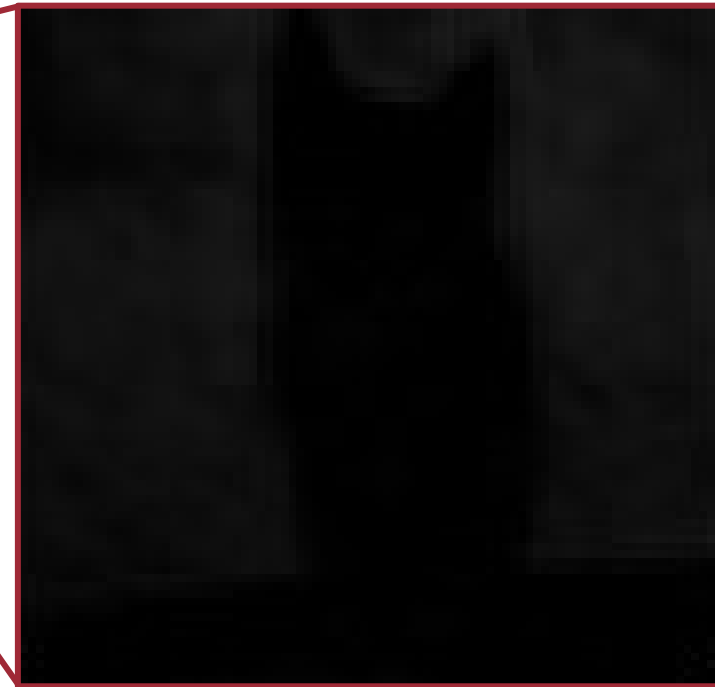


Activations in a Trained AlexNet

1st convolutional layer

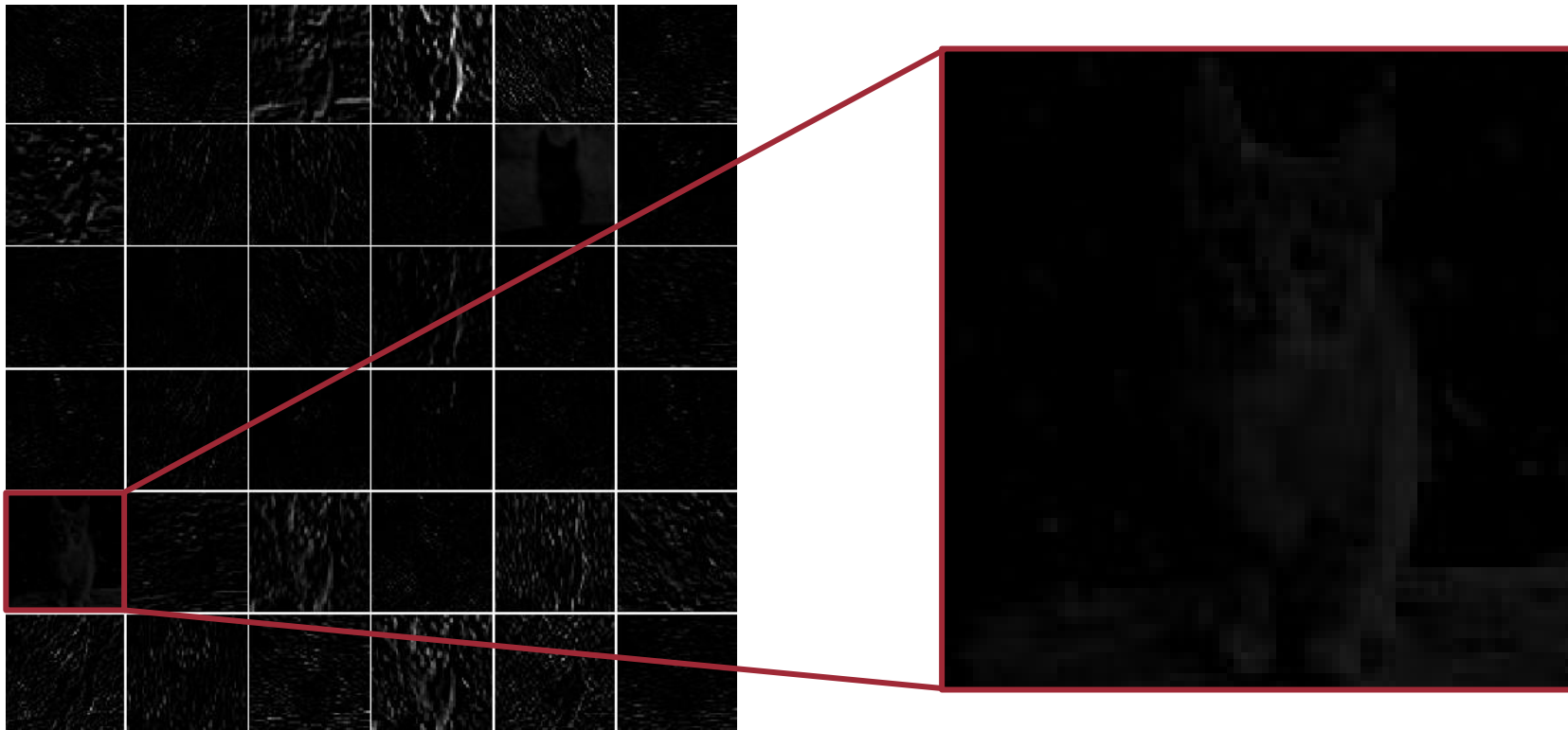


What's this!?



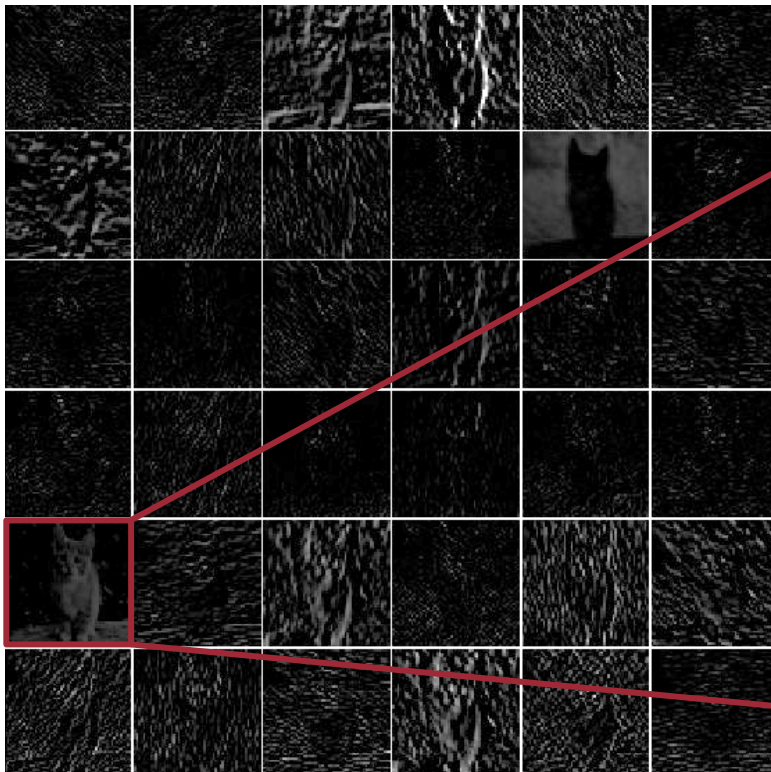
Activations in a Trained AlexNet

1st convolutional layer



Activations in a Trained AlexNet

1st convolutional layer



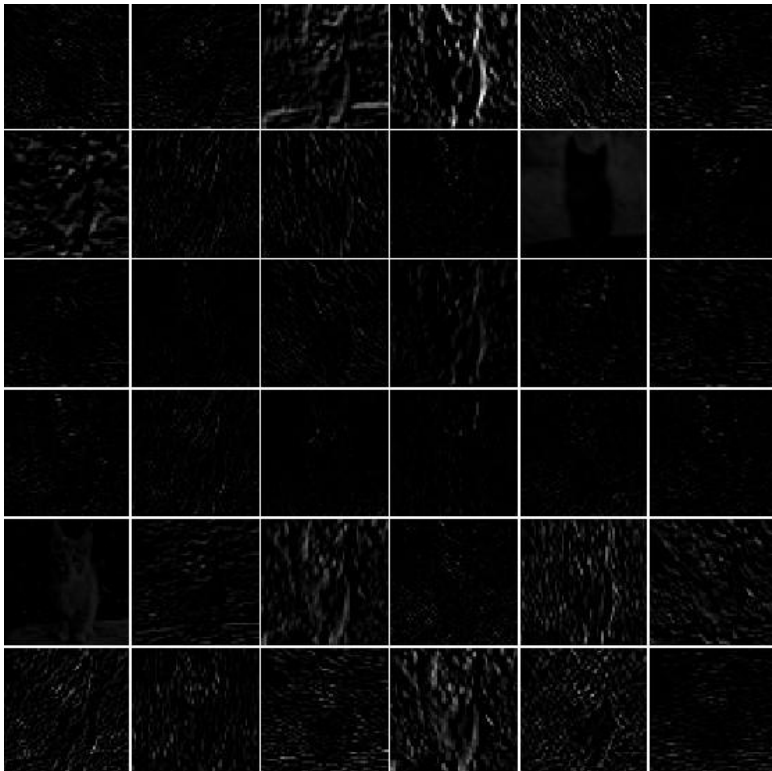
What's this!?



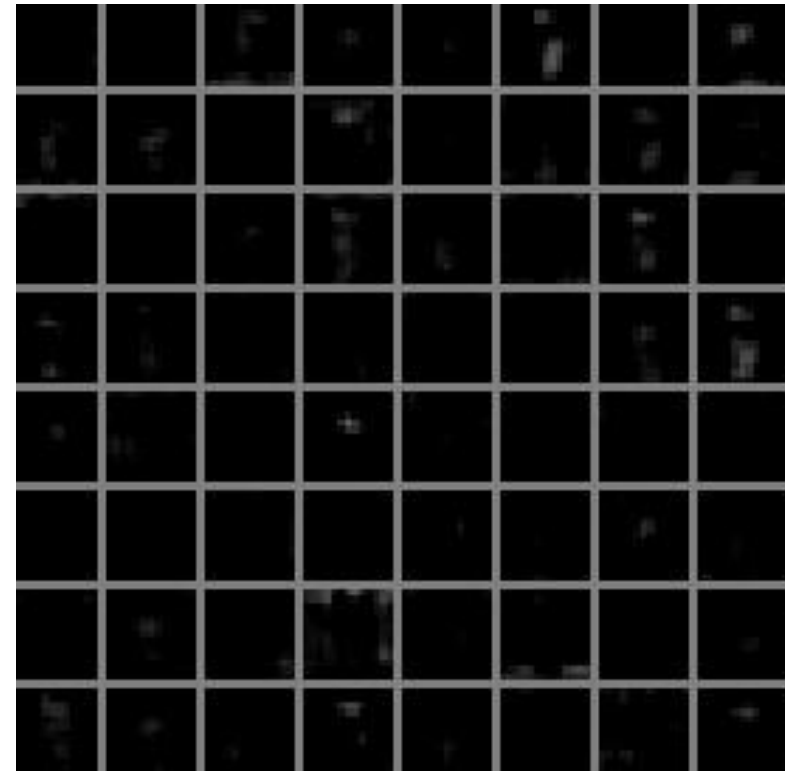
Brightness+
Contrast+

Activations in a Trained AlexNet

1st convolutional layer



5th convolutional layer

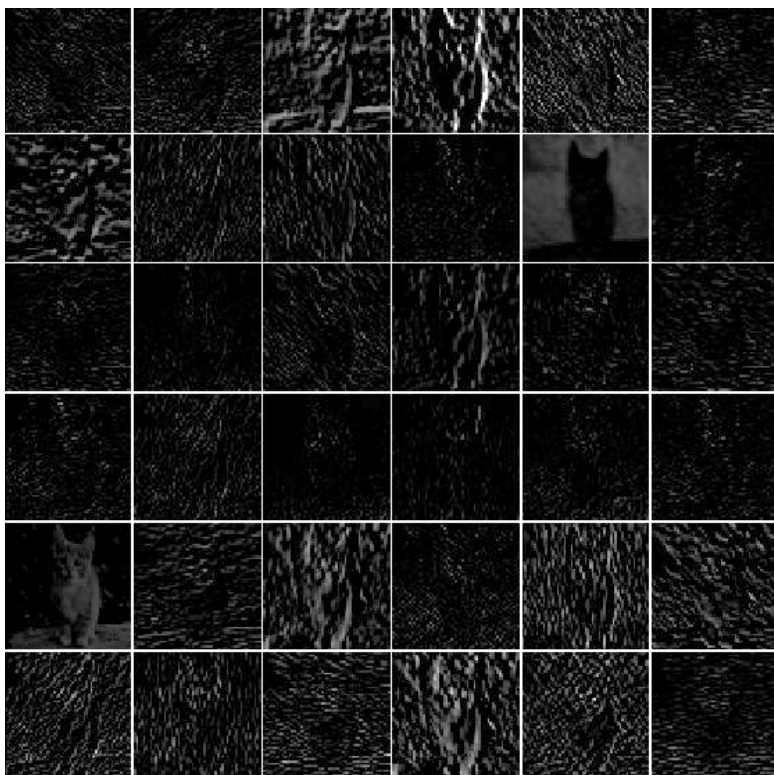


Activations in a Trained AlexNet

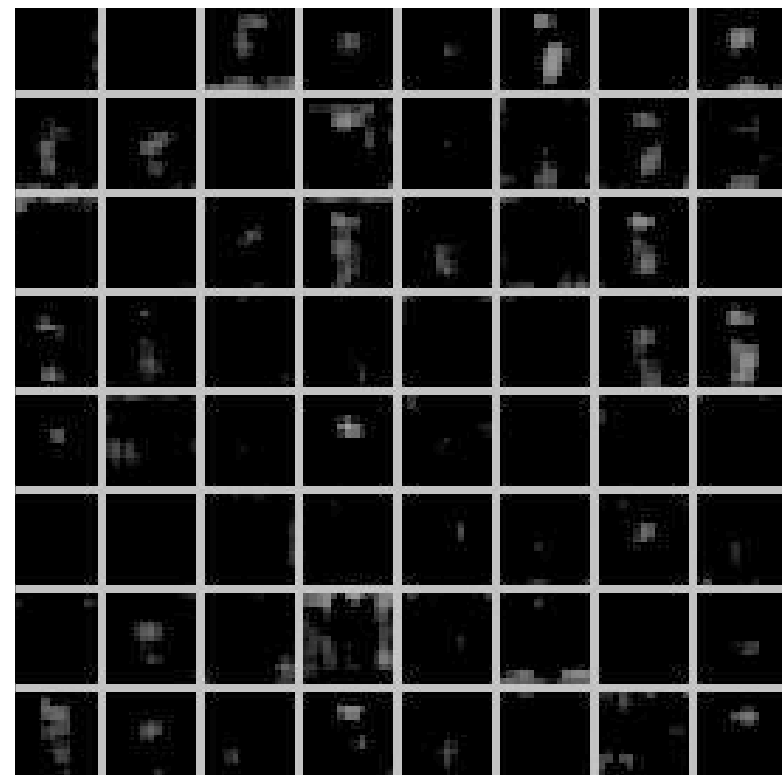
1st convolutional layer

5th convolutional layer

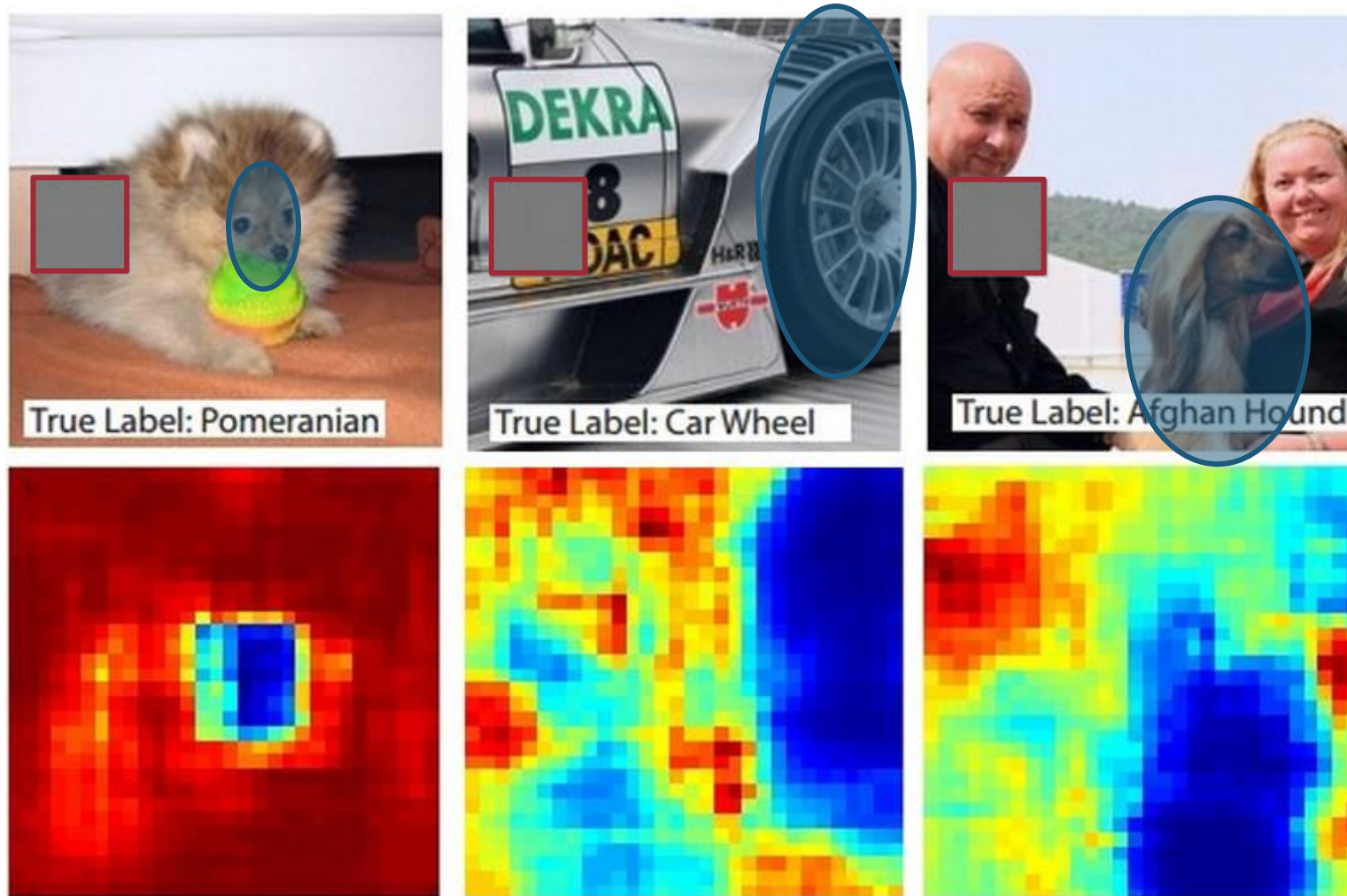
Brightness+
Contrast+



Brightness+
Contrast+



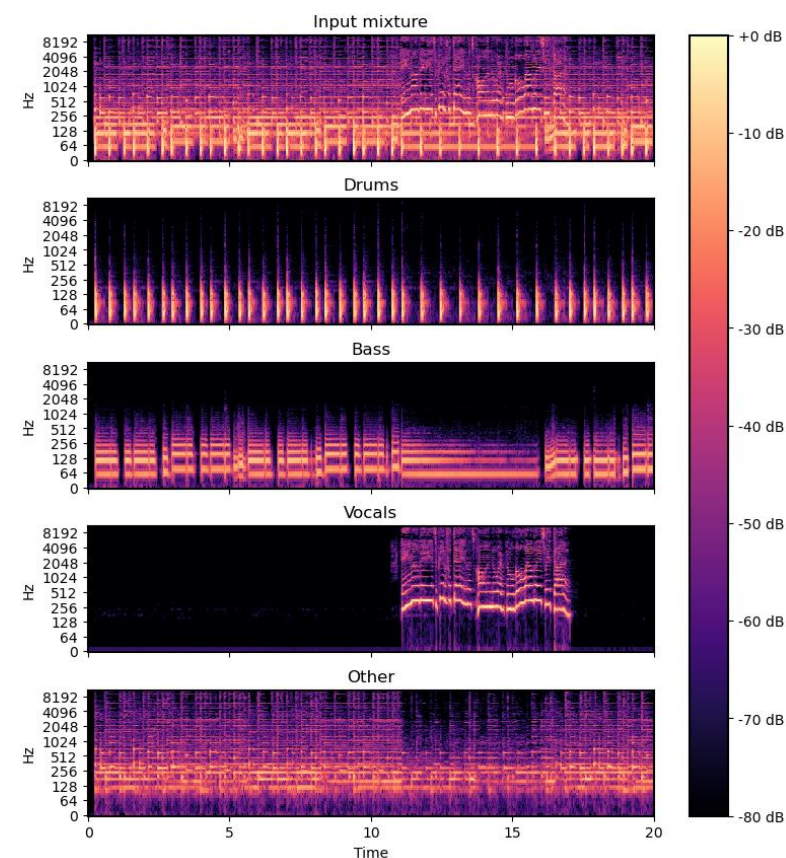
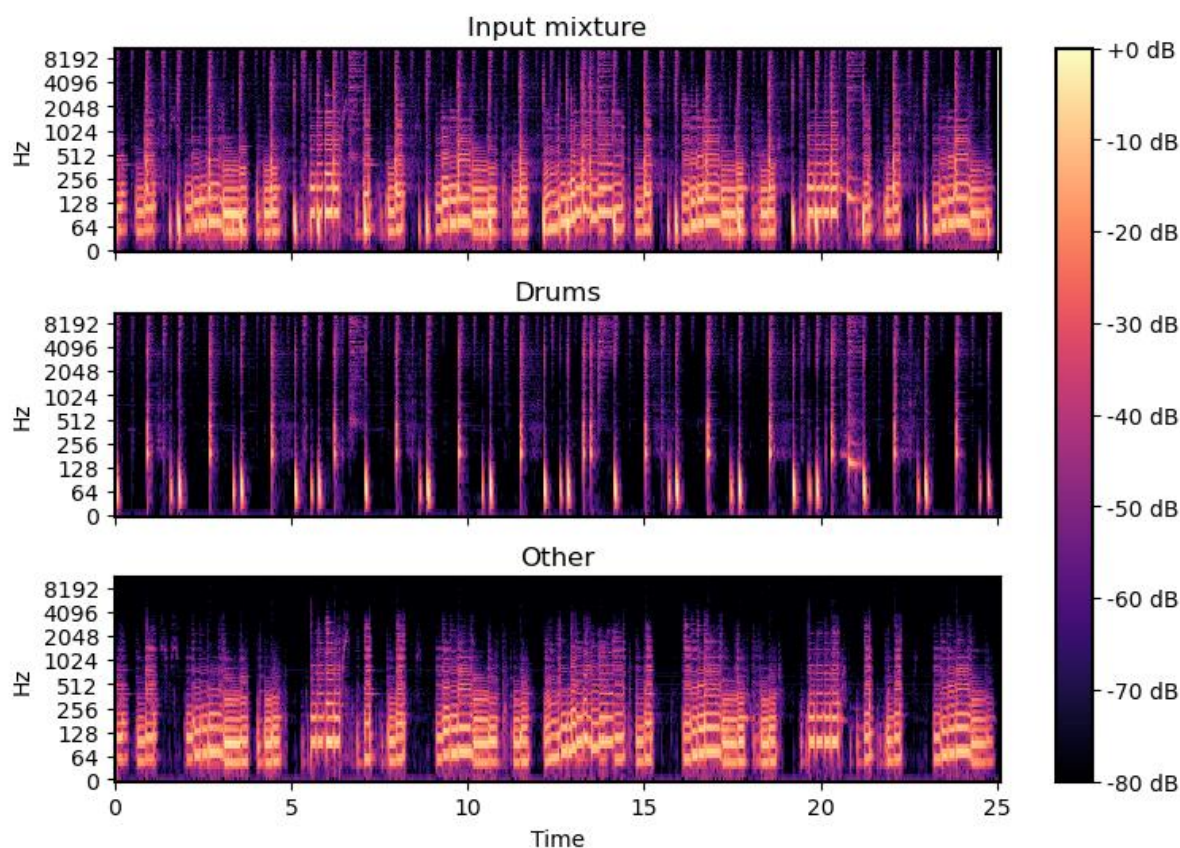
What does a CNN Learn?



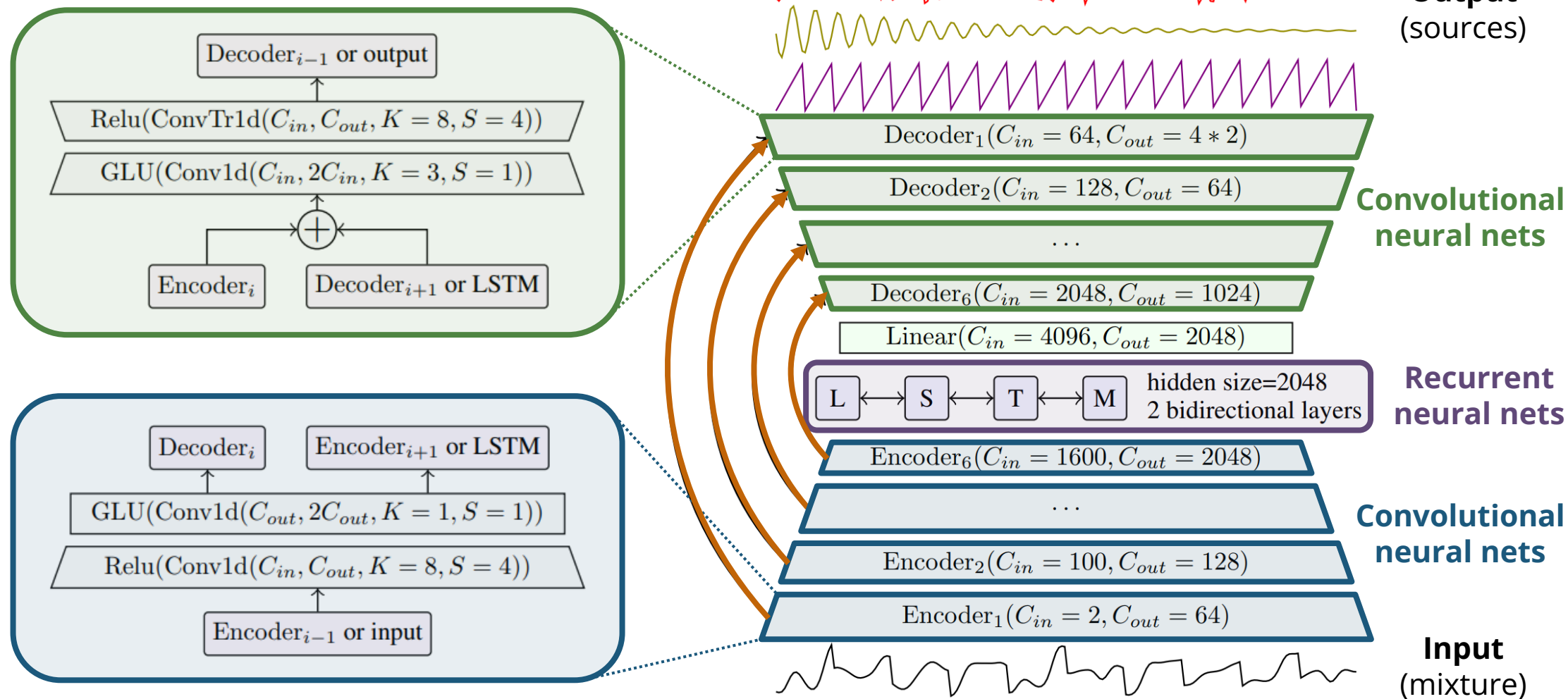
U-Nets

PA 3: Source Separation

- Part 2: Music Source Separation using Demucs

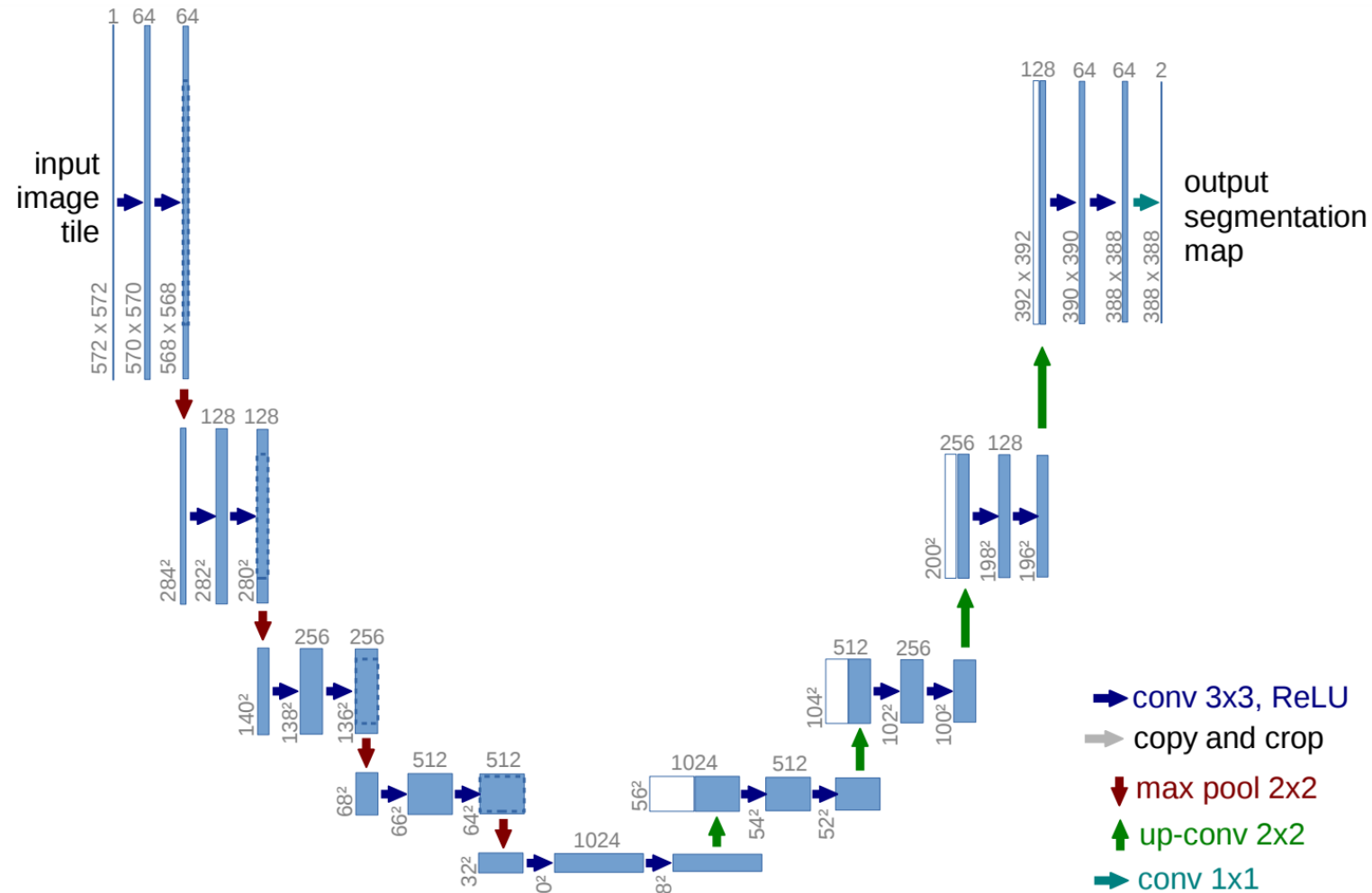


Demucs (Défossez et al., 2019)



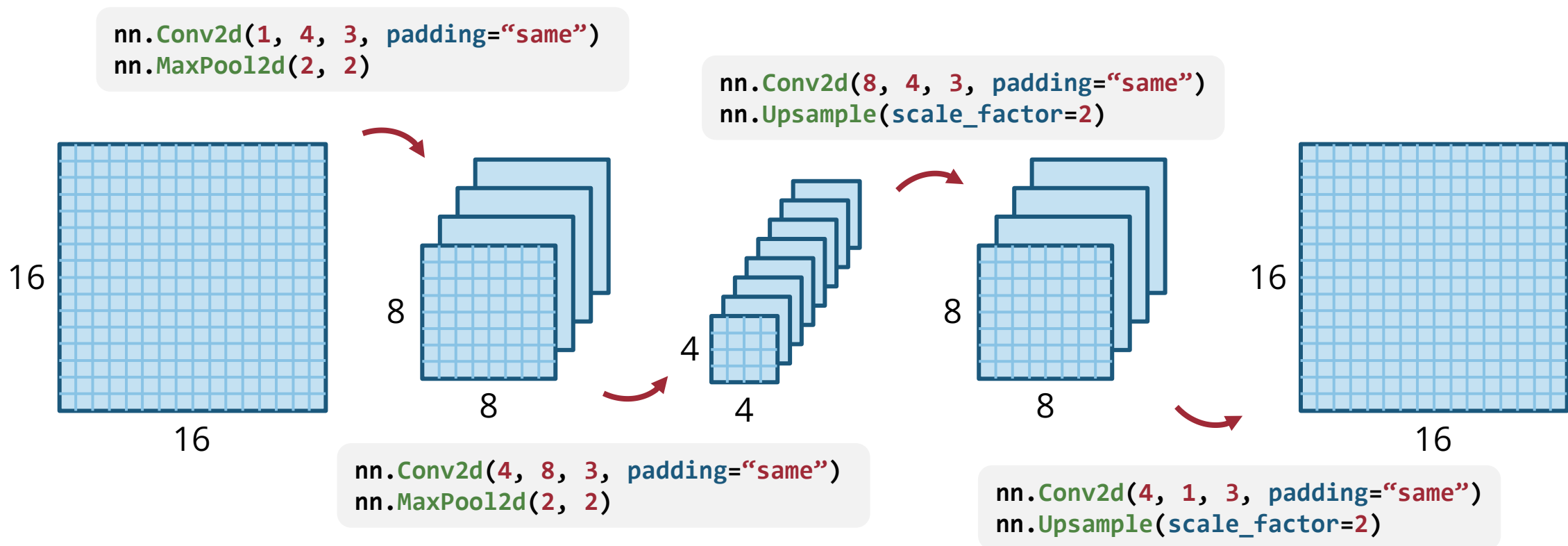
(Source: Défossez et al., 2019)

U-Nets (Ronneberger et al., 2015)

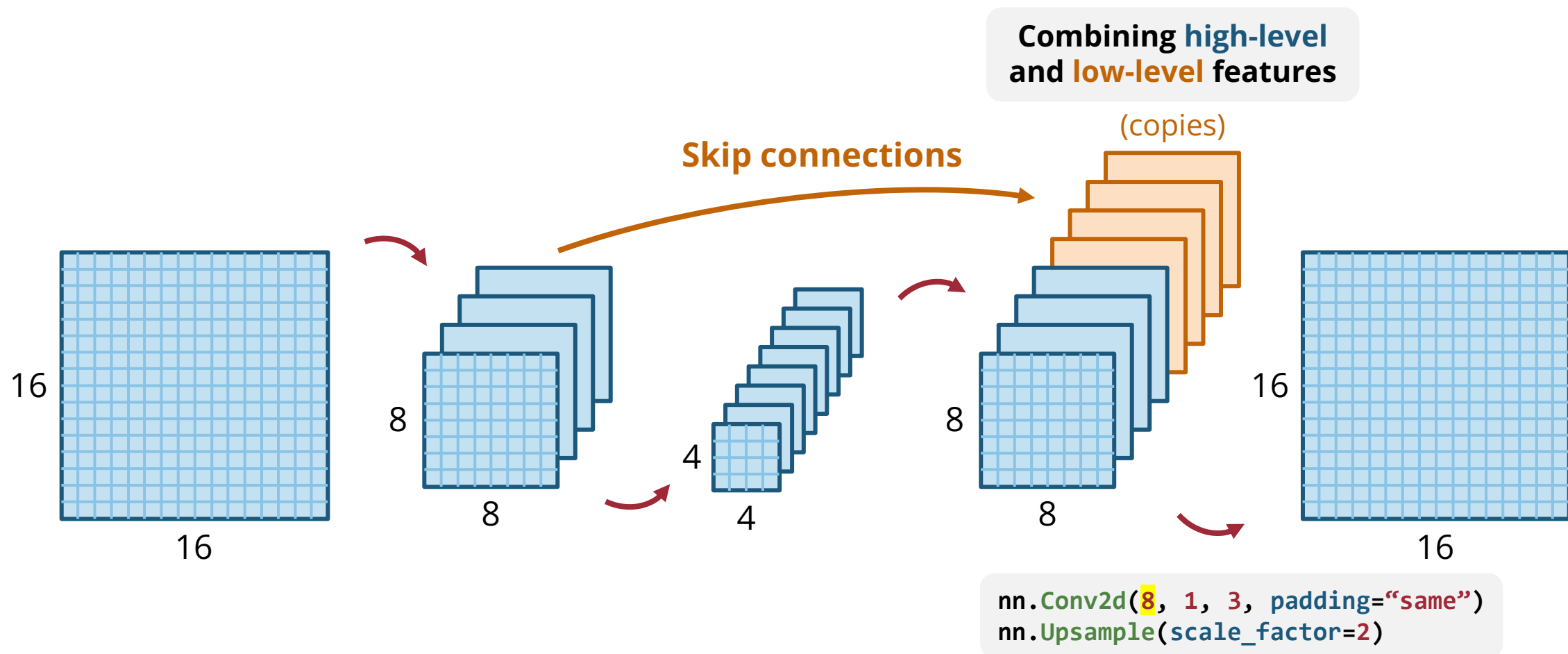


(Source: Ronneberger et al., 2015)

A Toy Example of a U-Net

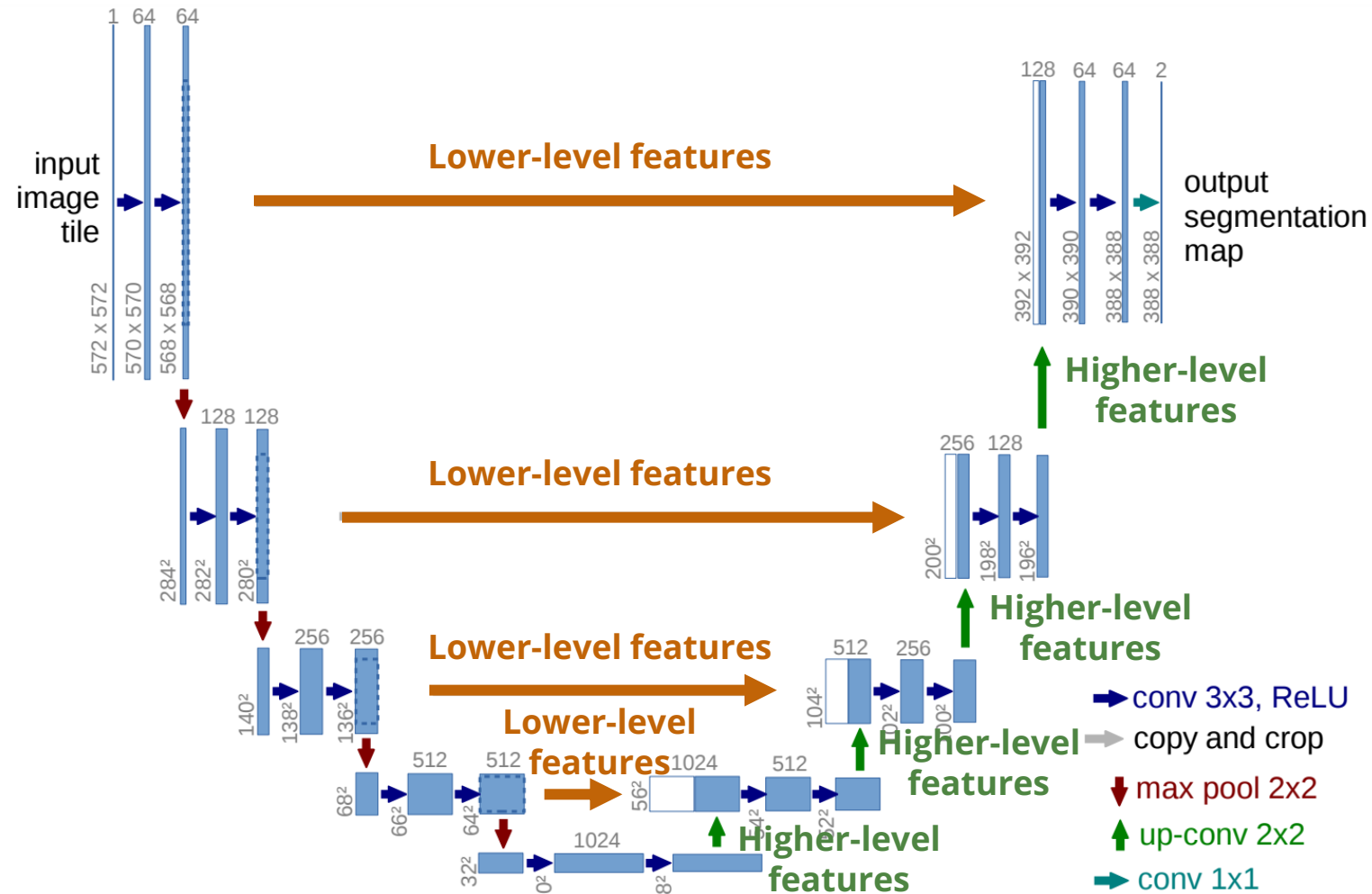


A Toy Example of a U-Net



U-Nets are useful when the inputs and outputs have the same shape!

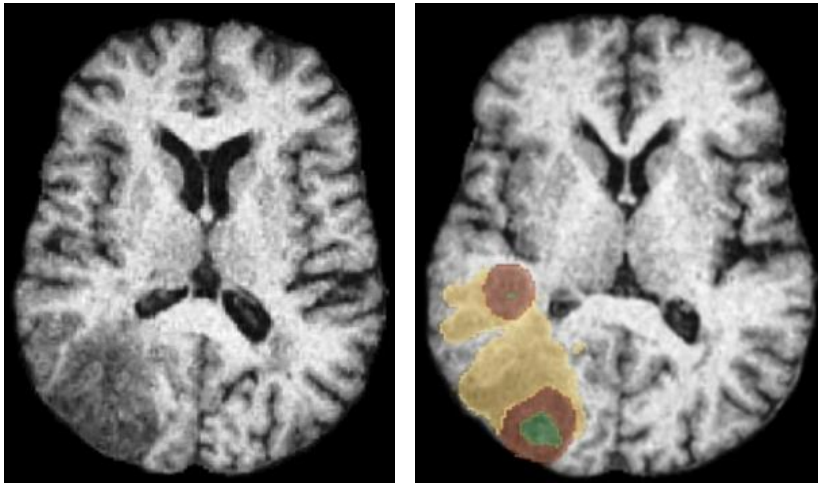
U-Nets (Ronneberger et al., 2015)



(Source: Ronneberger et al., 2015)

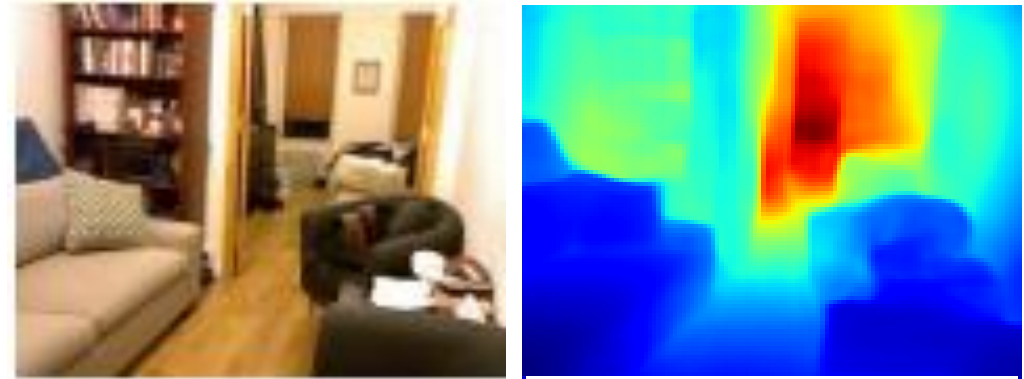
Applications of U-Nets

Tumor Segmentation



(Source: Kharaji et al., 2024)

Depth Estimation



(Source: Barakat, 2018)

Image Segmentation



(Source: Kirillov et al., 2023)

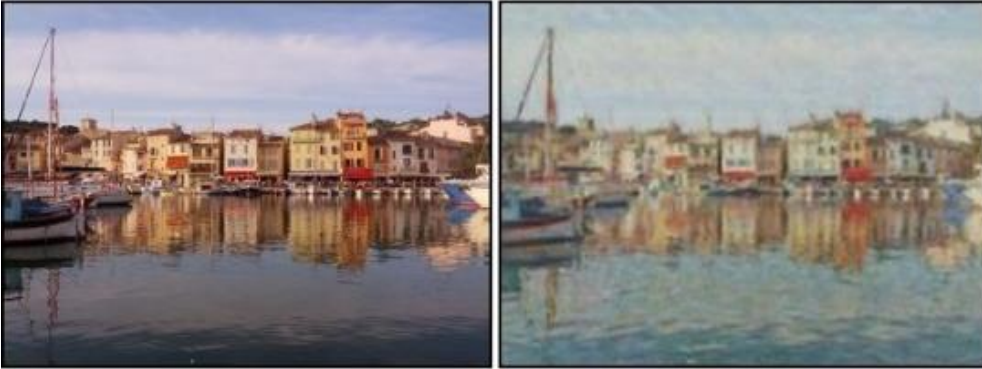
Omar Barakat, "Depth estimation with deep Neural networks part 1," *Medium*, January 11, 2018

Mona Kharaji, Hossein Abbasi, Yasin Orouskhani, Mostafa Shomalzadeh, Foad Kazemi, and Maysam Orouskhani, "nnU-Net for Brain Tumor Segmentation," *Neuroscience Informatics*, 2024.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick, "Segment Anything," *ICCV*, 2023.

Applications of U-Nets

Style Transfer



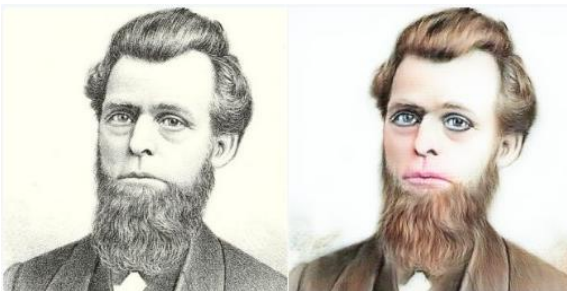
(Source: Zhu et al., 2018)

Sim2Real



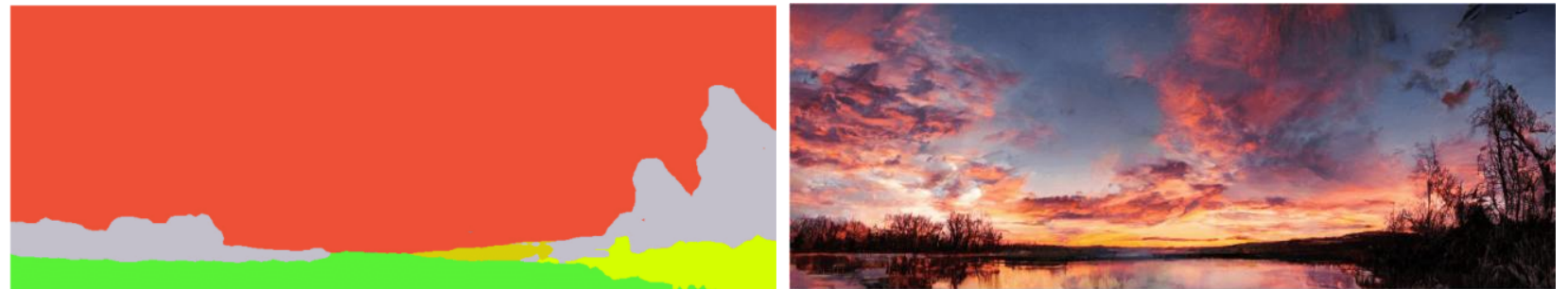
(Source: Zhu et al., 2018)

Colorization



(Source: Zhu et al., 2018)

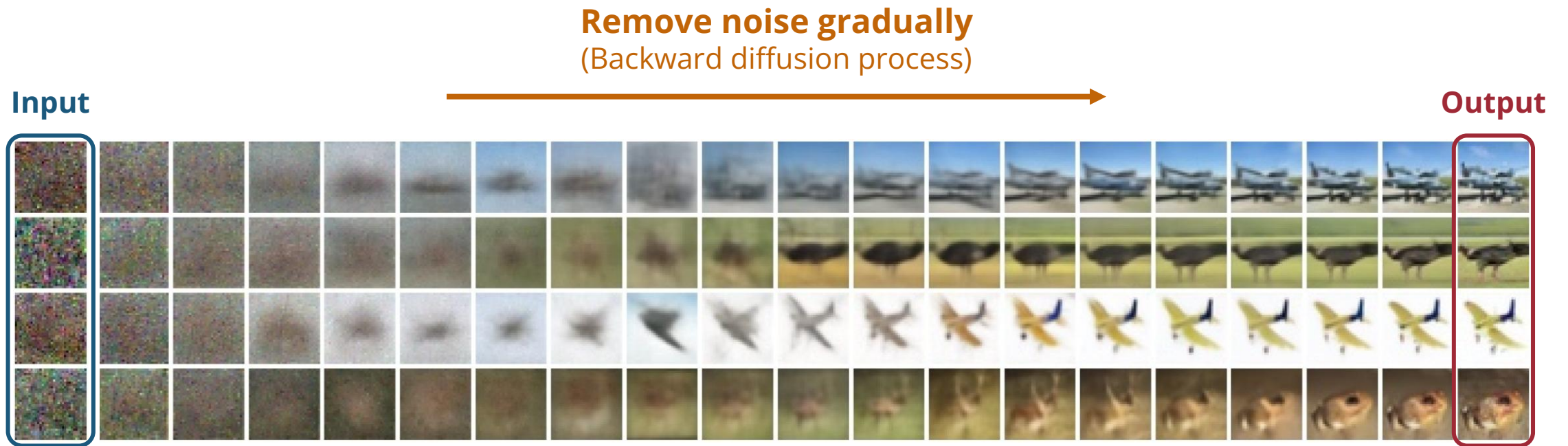
Semantic Synthesis



(Source: Rombach et al., 2022)

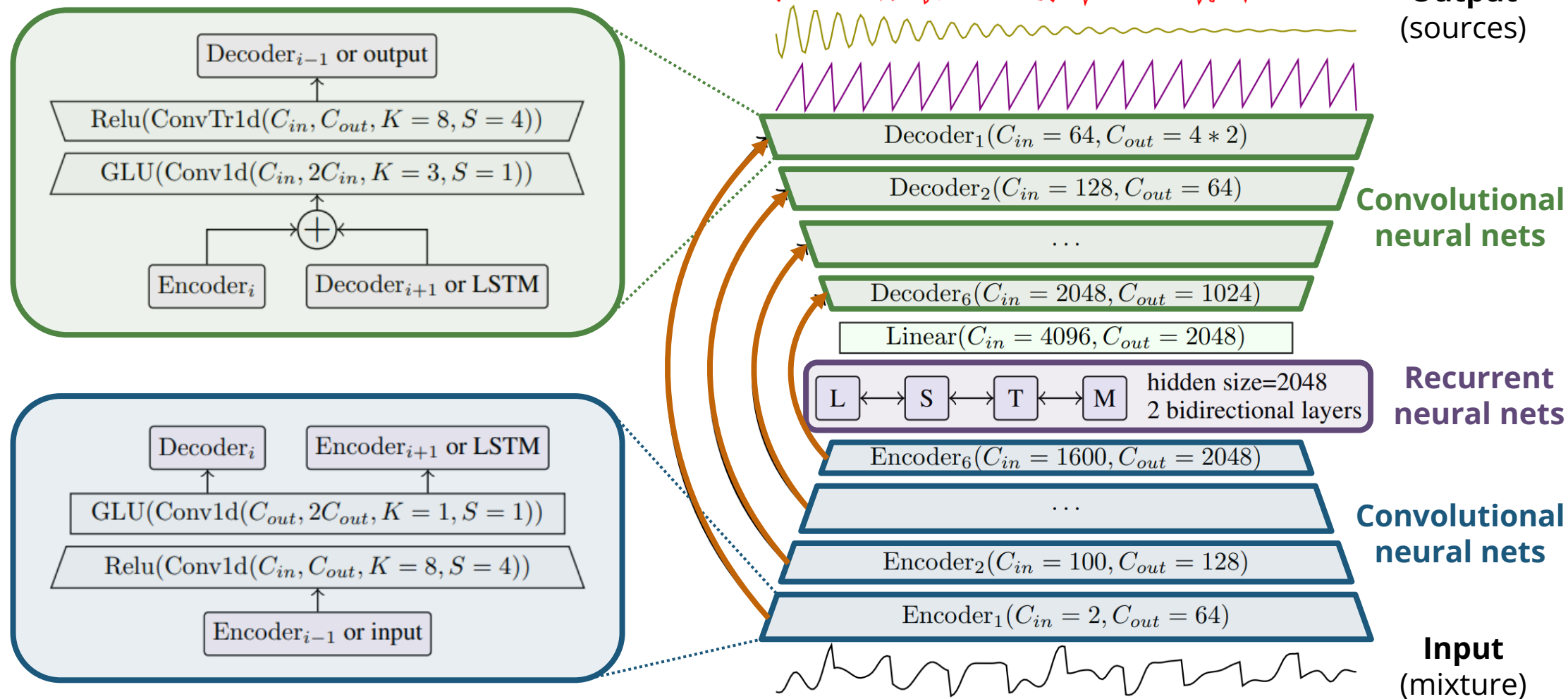
Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *CVPR*, 2022.
Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *ICCV*, 2017.

Applications of U-Nets: Diffusion Models



(Source: Ho et al., 2020)

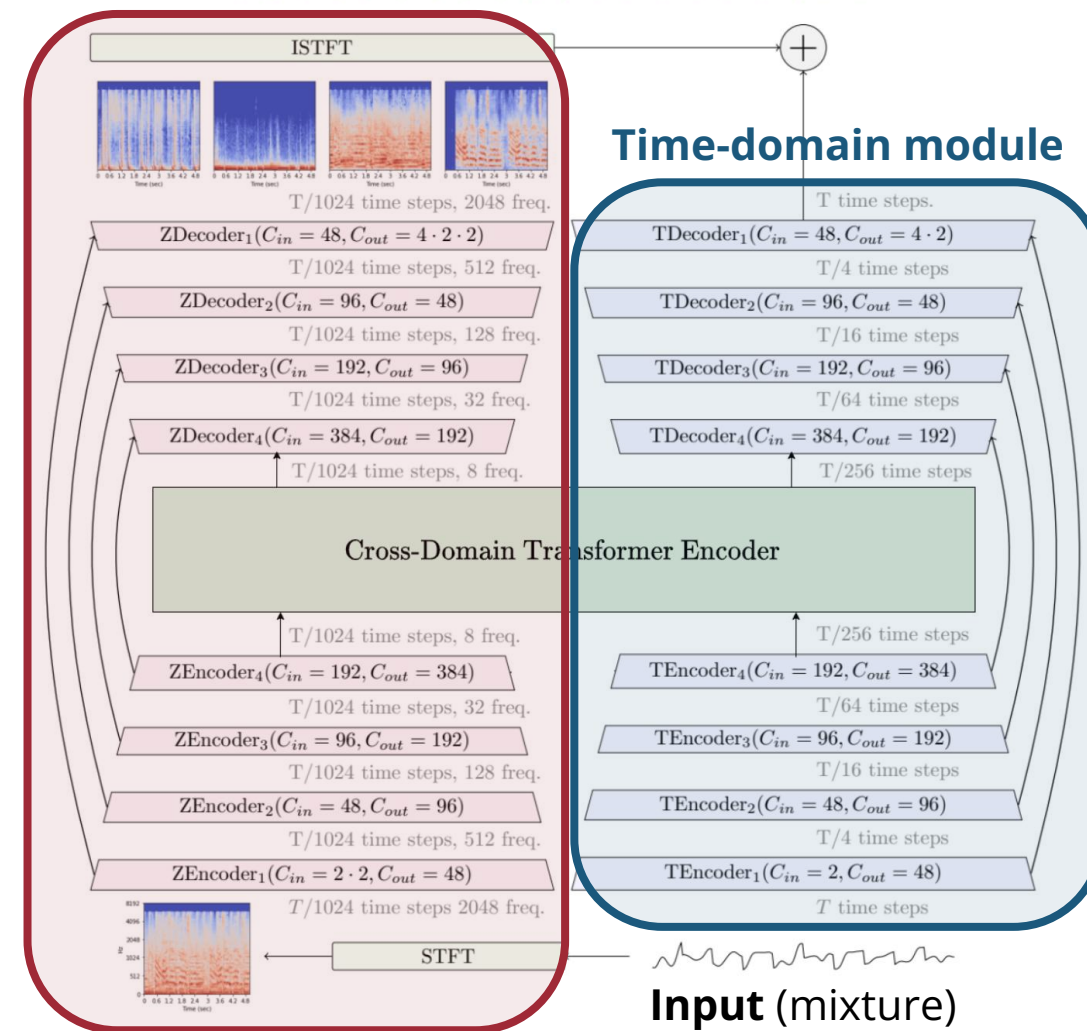
Demucs (Défossez et al., 2019)



(Source: Défossez et al., 2019)

Demucs-Hybrid (Rouard et al., 2023)

Frequency-domain module



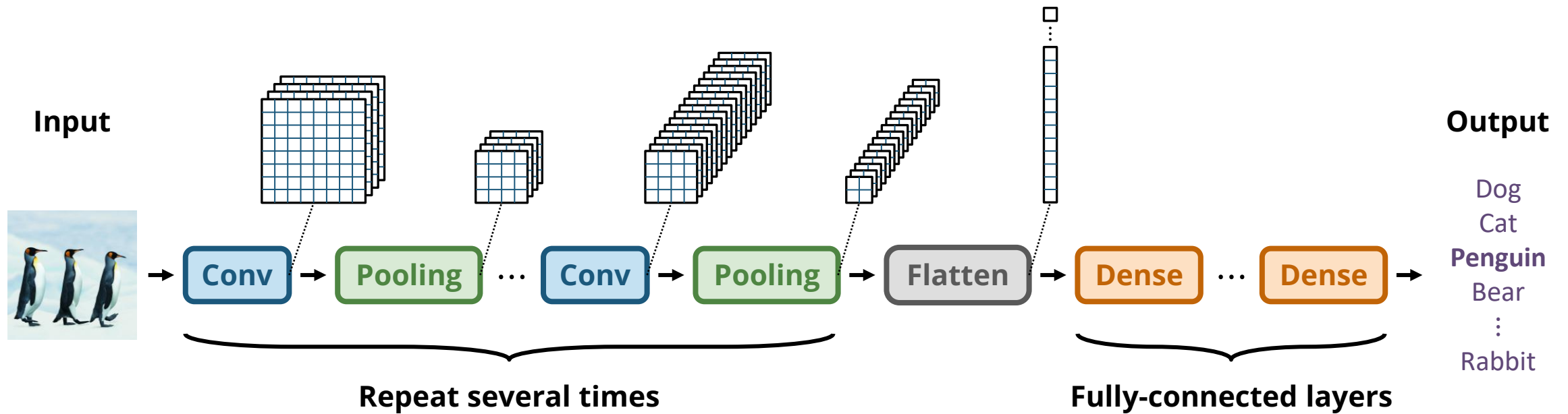
(Source: Rouard et al., 2023)

Recap

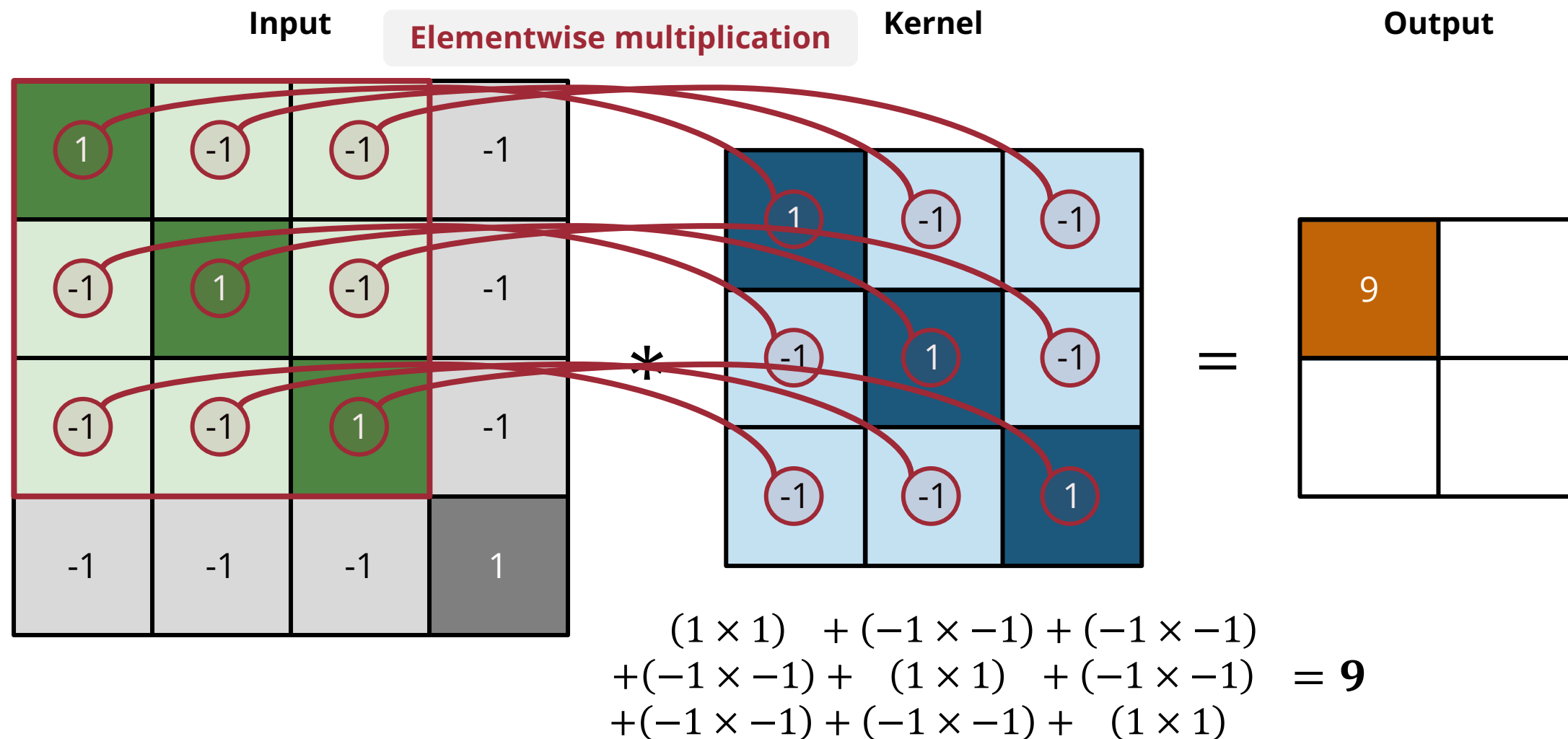
Reusable Pattern Detectors



Convolutional Neural Network (CNNs)

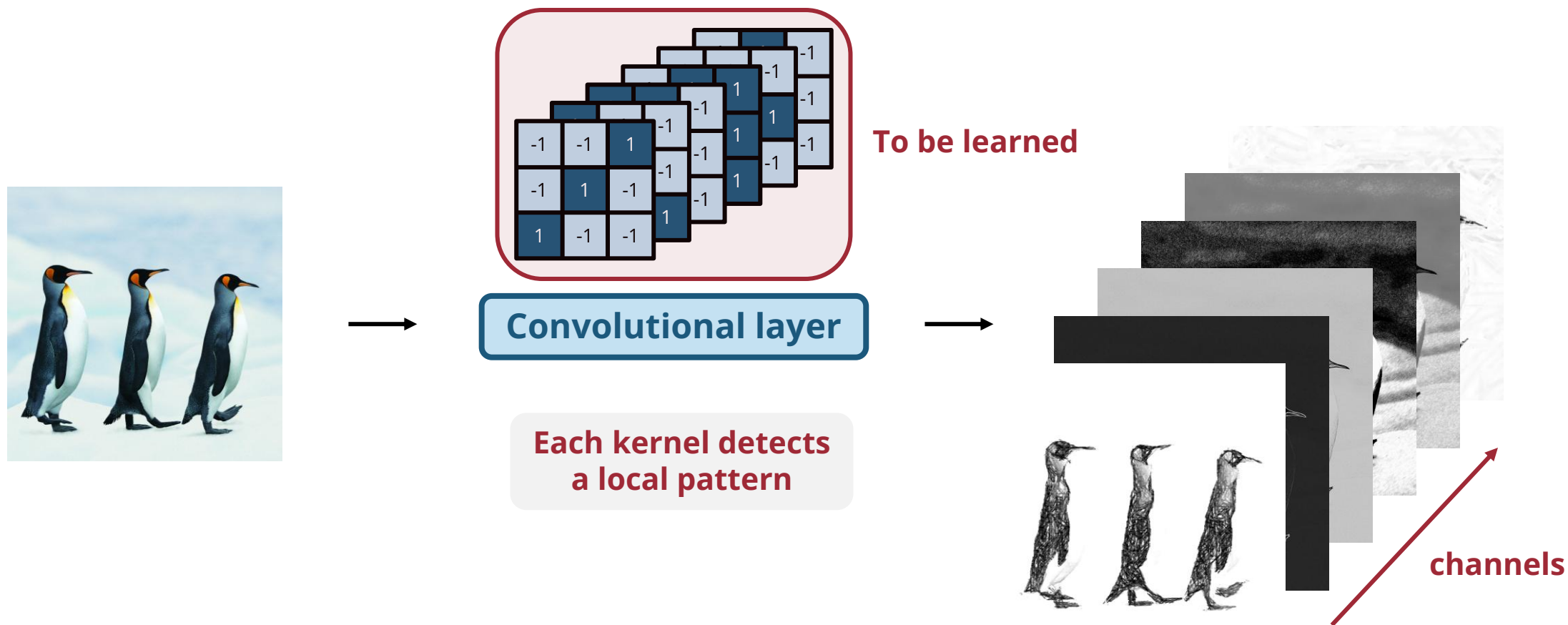


2D Convolution

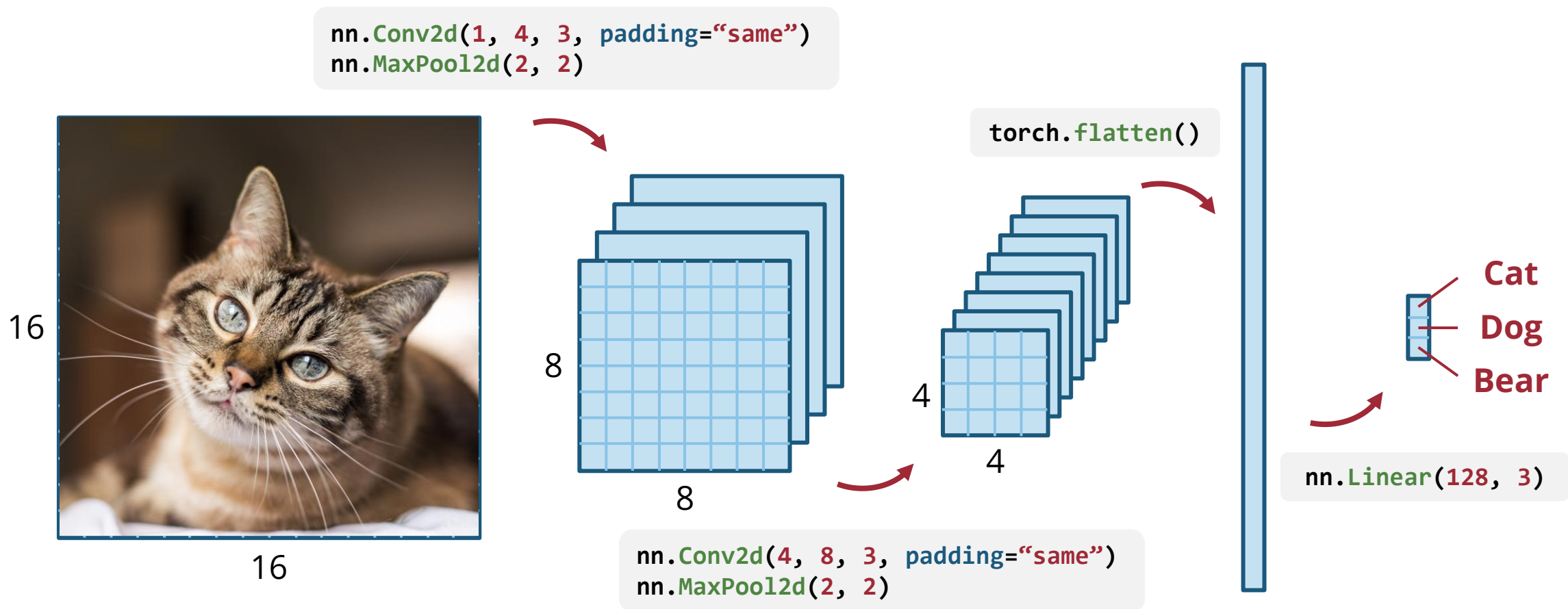


Convolutional Layer

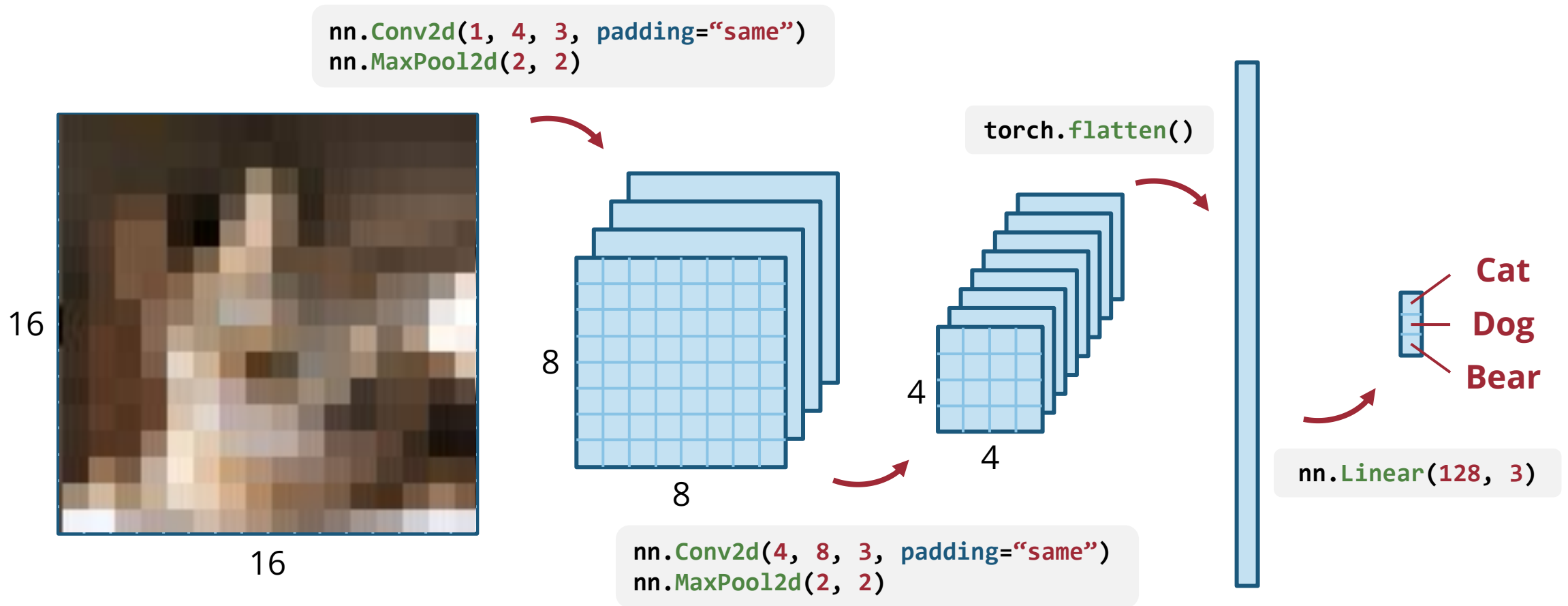
- A convolutional layer consists of many **learnable kernels** (channels)



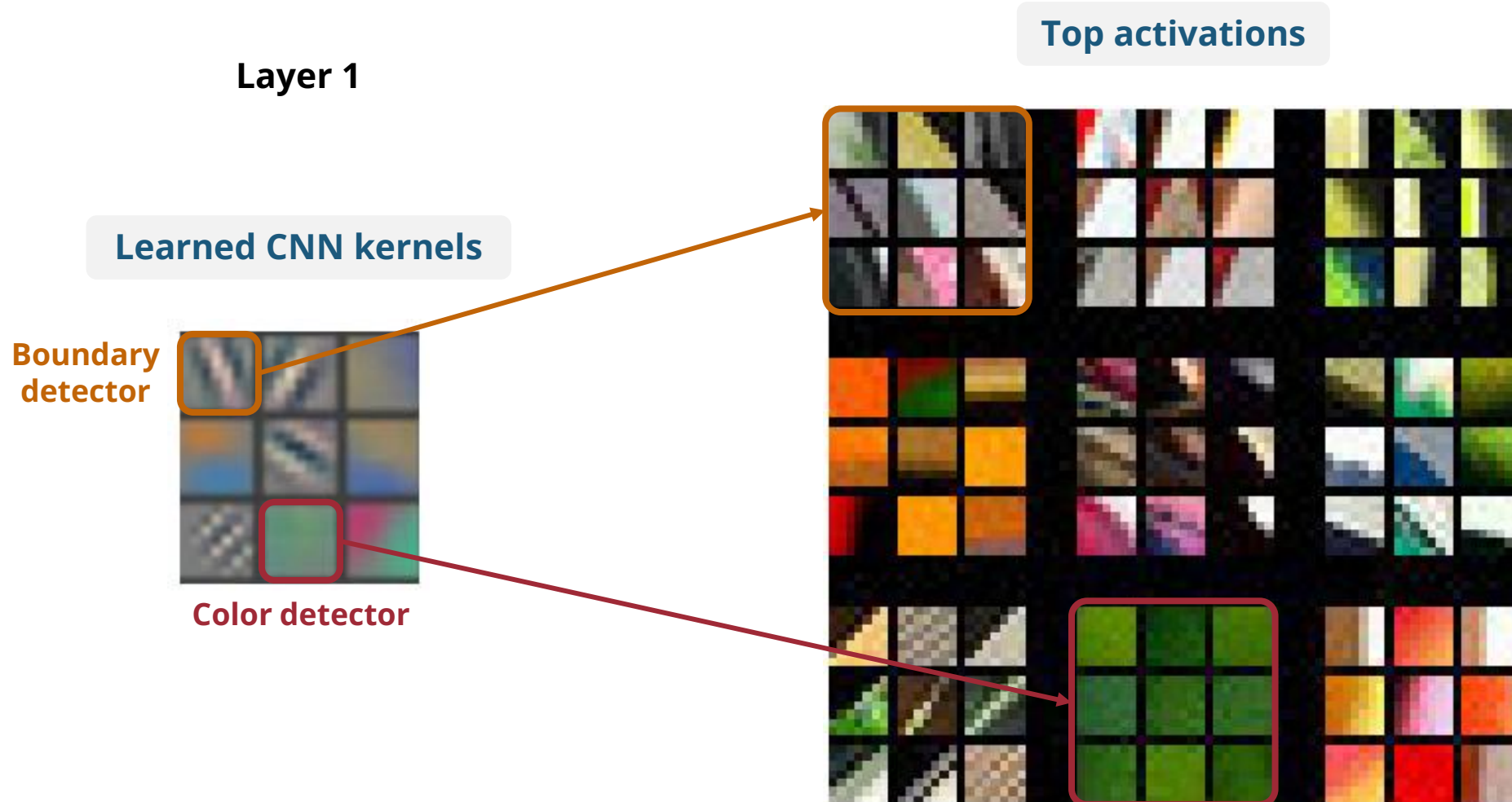
A Toy Example



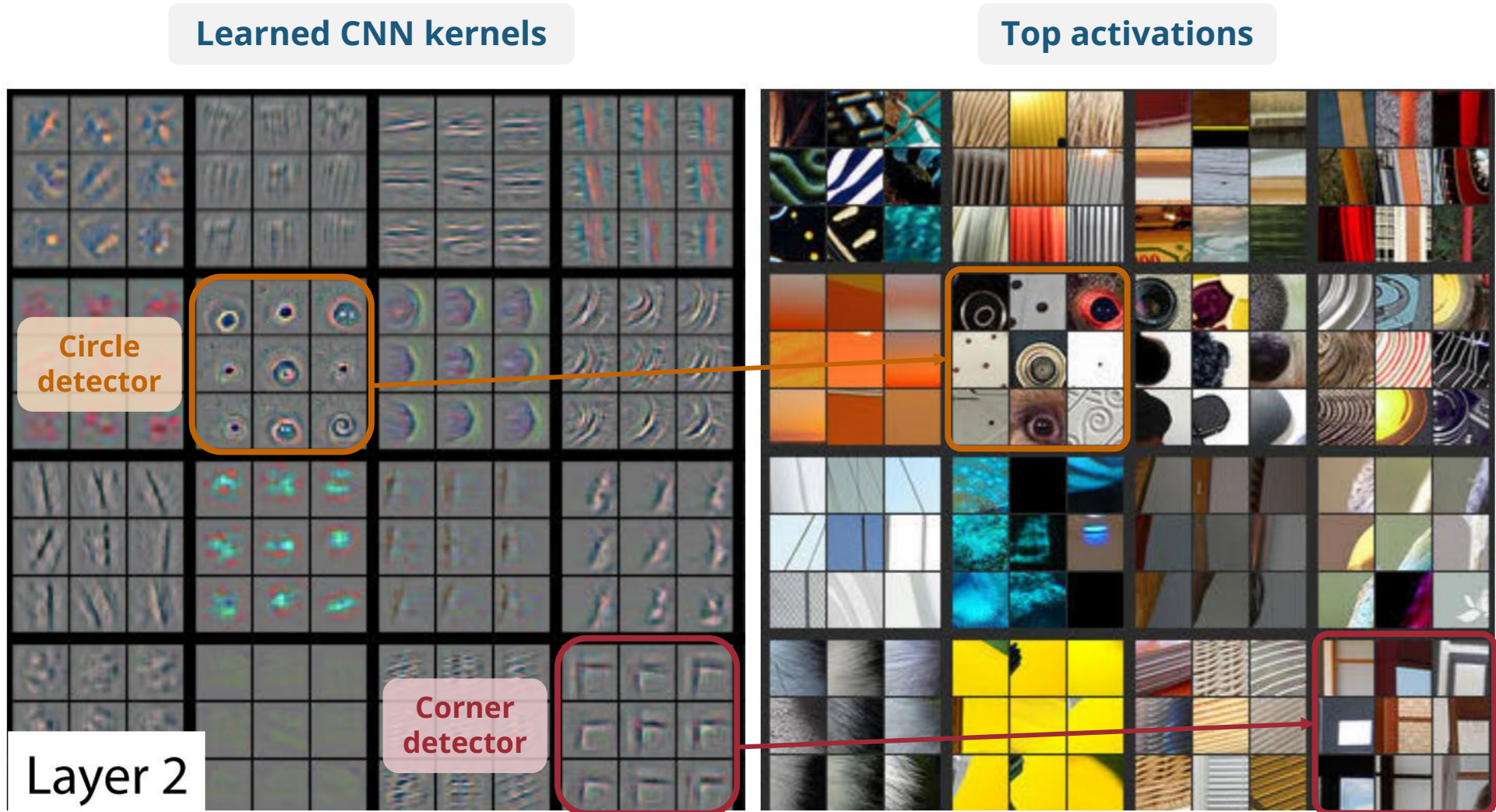
A Toy Example



Learned CNN Kernels in a Trained AlexNet



Learned CNN Kernels in a Trained AlexNet

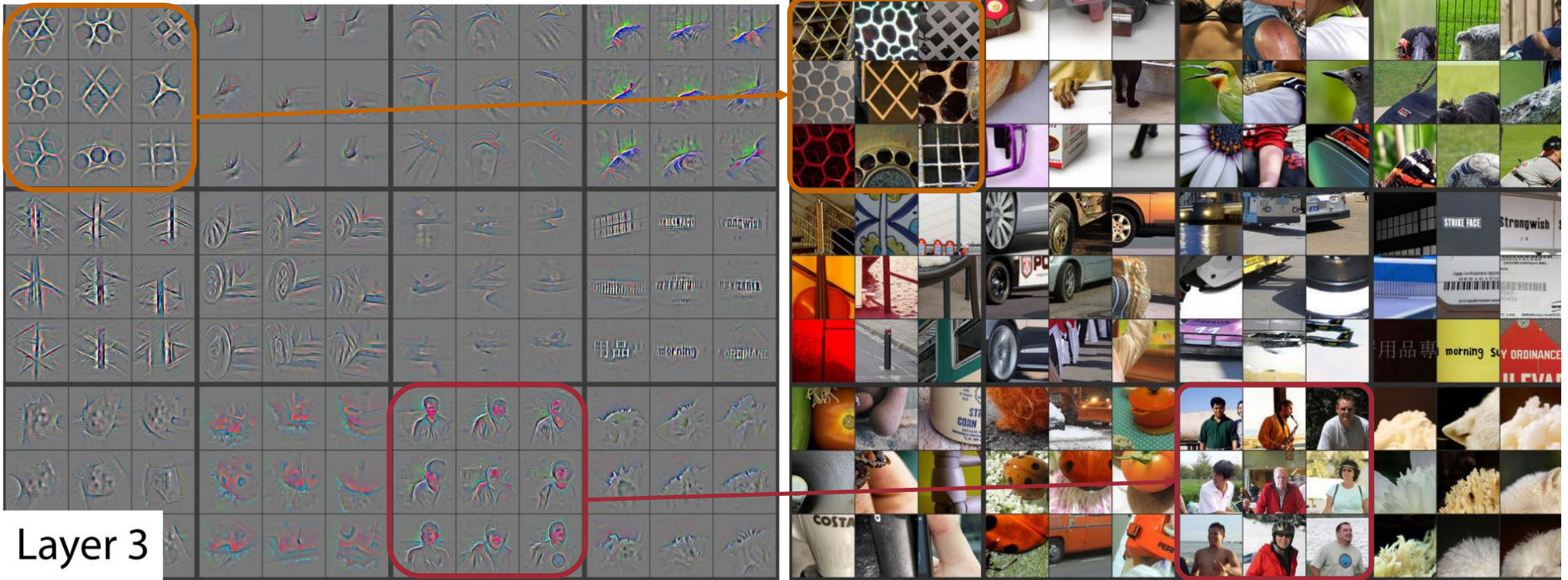


Learned CNN Kernels in a Trained AlexNet

Learned CNN kernels

Top activations

Grid detector

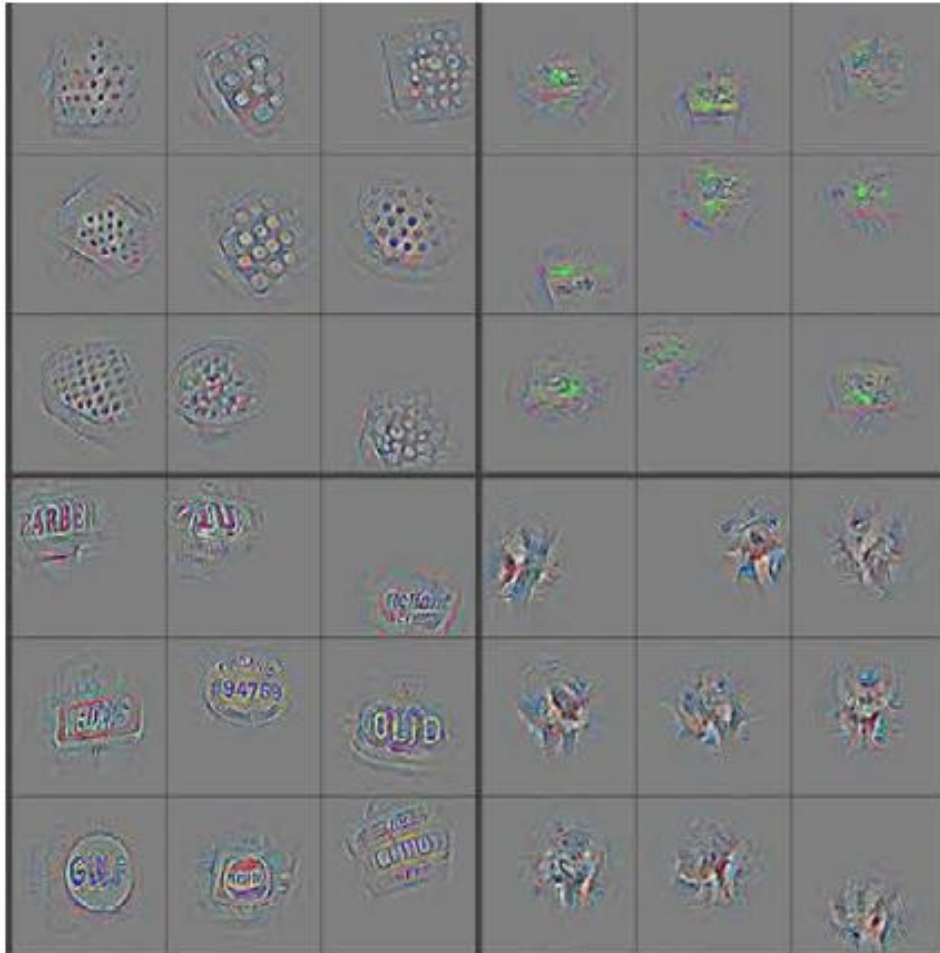


Layer 3

Human detector

Learned CNN Kernels in a Trained AlexNet

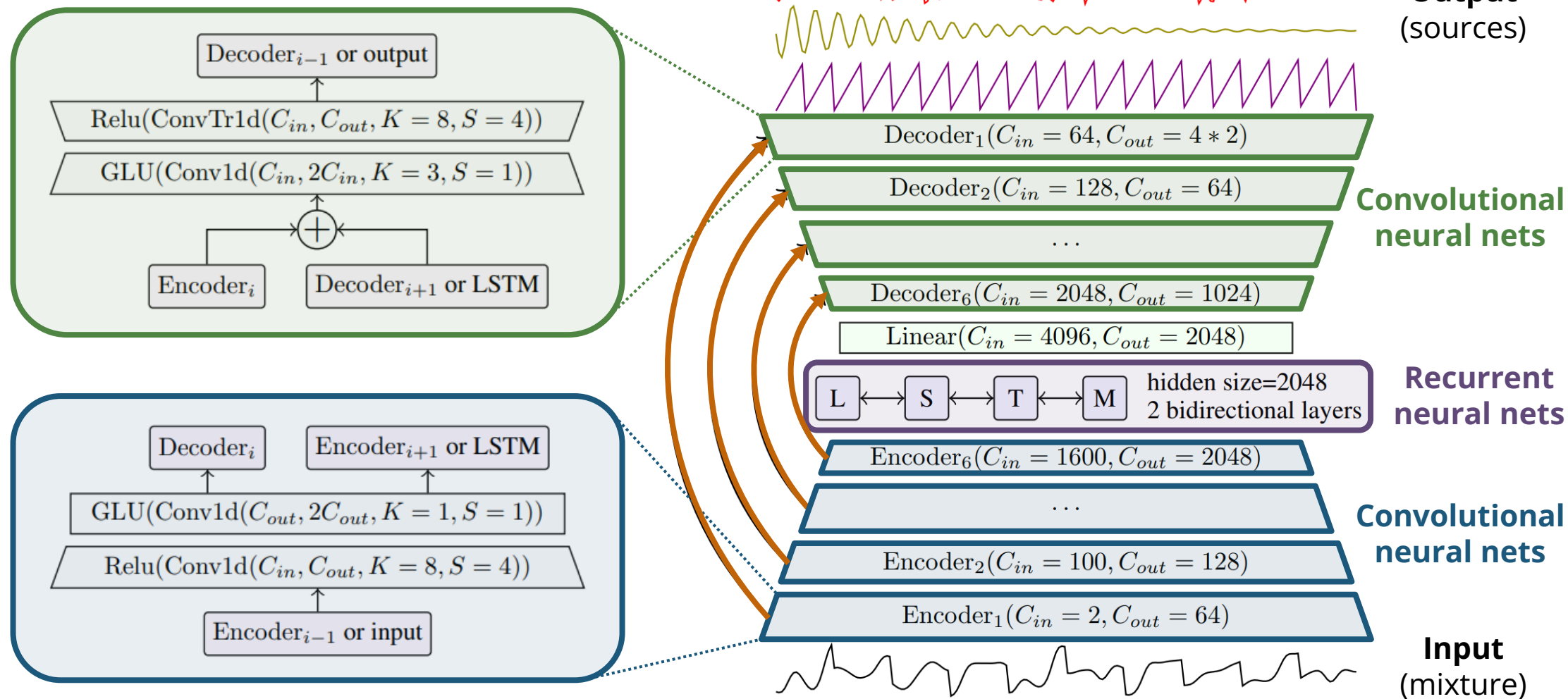
Learned CNN kernels



Top activations

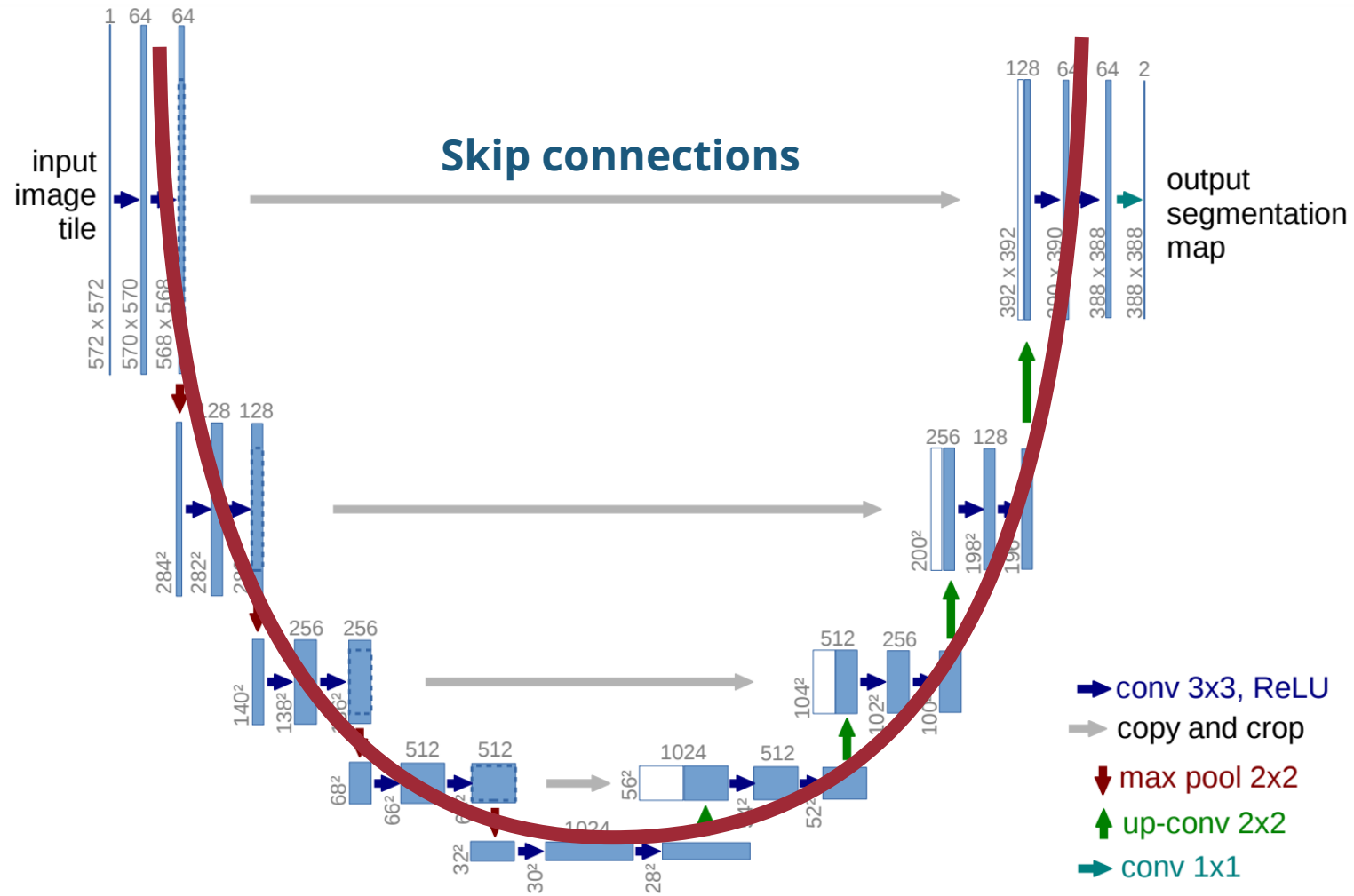


Demucs (Défossez et al., 2019)



(Source: Défossez et al., 2019)

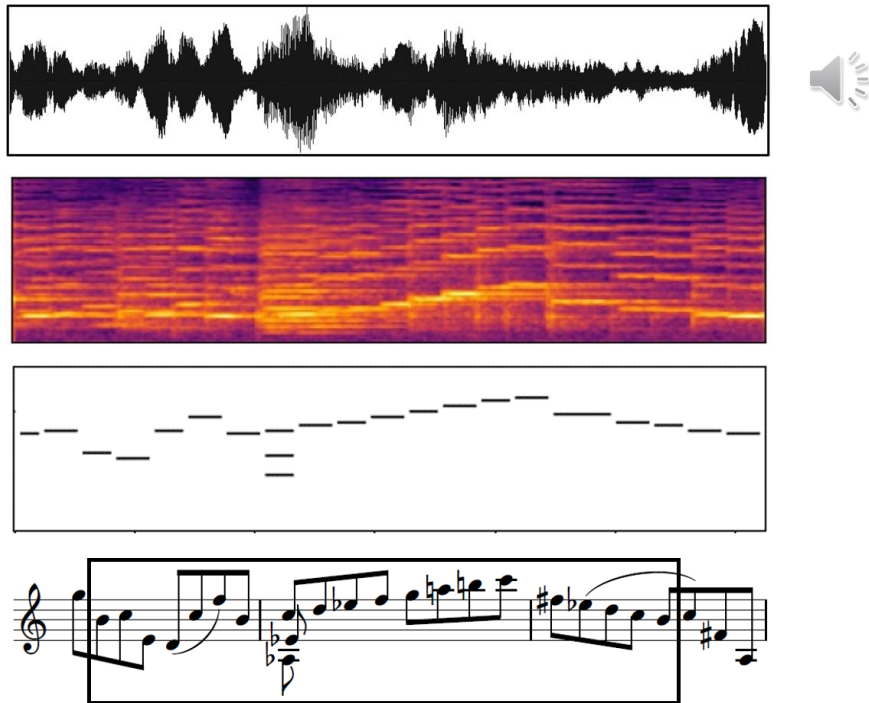
U-Nets (Ronneberger et al., 2015)



(Source: Ronneberger et al., 2015)

Next Lecture

Music Transcription & Analysis



(Source: Dong et al., 2022)